hyväksymispäivä arvosana

arvostelija

Lasten Areenan suosittelu

Tuukka Paukkunen

Helsinki 14.2.2019

Pro gradu -tutkielma

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Matemaattis-luonnontieteellinen tiedekunta	Tietojenkäsittelytieteen laitos

Tekijä — Författare — Author
Tuukka Paukkunen

Työn nimi — Arbetets titel — Title

Lasten Areenan suosittelu

Oppiaine — Läroämne — Subject
Tietojenkäsittelytiede

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Pro gradu -tutkielma	14.2.2019	78 sivua ja 4 liitesivua

Tiivistelmä — Referat — Abstract

Sisältöjen automaattinen suosittelu on oleellinen osa nykyaikaista verkkomediapalvelua. Ohjelmien automaattinen suosittelu on merkittävässä asemassa myös Yle Areenassa. Yle Areenan tarjonta koostuu useista kymmenistä tuhansista ohjelmista, joten on perusteltua, että käyttäjää autetaan löytämään hänelle relevanttia sisältöä. Verkkomediapalvelujen käyttäjäkunnat koostuvat joukosta yksilöitä. Sen vuoksi sisältöjen suosittelun täytyy mukautua kunkin käyttäjän yksilöllisiin tarpeisiin.

Sisältöjen suosittelumenetelmät jaetaan yhteistoiminnallisiin suosittelumenetelmiin (engl. collaborative filtering), sisältöön perustuviin suosittelumenetelmiin (engl. content-based recommendation) tai jonkinlaiseen edellä mainittujen yhdistelmään. Yhteistoiminnalliset suosittelumenetelmät jaetaan edelleen muistipohjaisiin (engl. memory-based) suosittelumenetelmiin ja mallipohjaisiin (engl. model-based) suosittelumenetelmiin.

Yleisradio tarjoaa lapsille oman verkkomediapalvelun, Lasten Areenan. Lasten Areenassa ei tällä hetkellä suositella sisältöjä automaattisesti, vaan sisällöt järjestetään medioiden julkaisuajankohdan perusteella. Lastenohjelmia on myös tarjolla runsaasti, jolloin lapset ovat aikuisten tapaan saman haasteen äärellä: miten löytää sisältövalikoimasta itselle mieluista tai hyödyllistä sisältöä.

Tässä tutkielmassa selvitetään, minkälainen toteutustapa tulisi valita Lasten Areenan sisältöjen automaattiseen suositteluun. Tutkimuksessa tarkastellaan Areenan ohjelmien toistotietojen erityispiirteitä sekä vertaillaan erilaisia sisältöjen suosittelumenetelmiä. Tutkielmassa mitataan Areenan olemassa olevan suosittelujärjestelmän soveltuvuutta Lasten Areenan sisältösuositusten tuottamiseen. Lisäksi tutkimusta varten on toteutettu muistipohjainen suosittelujärjestelmä. Tutkielmassa mitataan myös sen tuottamien suositusten soveltuvuutta Lasten Areenan sisältösuosituksiksi. Soveltuvuutta mitataan herkkyys-mittarilla (engl. Recall), normalisoidulla diskontatulla kumulatiivisella arvolla (engl. Normalized Discounted Cumulative Gain) sekä monimuotoisuutta mittaavalla käänteisellä Simpson-indeksillä (engl. Inverse Simpson Index). Käänteisen Simpson-indeksin käyttö on suosittelualgoritmien yhteydessä harvinaista, mutta laissa määritellystä Yleisradion tehtävästä johtuen sen käyttö on perusteltua.

Tutkimuksessa saatujen tulosten perusteella Areenan olemassa oleva suosittelujärjestelmä soveltuu nykyisessä muodossaan parhaiten myös Lasten Areenan sisältöjen automaattiseksi suosittelumenetelmäksi.

ACM Computing Classification System (CCS):

- Information systems → Recommender systems
- Computing methodologies → Factorization methods

Avainsanat — Nyckelord — Keywords
Koneoppiminen, suosittelualgoritmit

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Esipuhe ja kiitokset

Työskentelen Yleisradiossa Yle Areenan kehityksen parissa. Tiimissämme esiintyi tarve tutkia lastenohjelmien automaattiseen suositteluun liittyviä piirteitä. Tavoitteena oli saada taustatietoa, jonka pohjalta Ylen Lasten Areenaan voitaisiin toteuttaa automattinen sisältöjen suositteluominaisuus. Tämän Pro gradu -tutkielman työstäminen oli osa tätä lastenohjelmien suosittelun tutkimustyötä.

Haluan kiittää tutkielman ohjauksesta työtoveriani Timo Ahoa. Timon kokemus ja asiantuntemus oli suureksi avuksi tutkimuksen suunnittelussa ja toteutuksessa. Timon ammattimainen tutkielmatekstin kommentointi oli myös korvaamatonta. Asiakkuuspäällikkö Susanna Snellin asiantuntemus lastenohjelmien kohderymästä sekä tuki tutkimuksen aikana oli myös olennaisessa osassa. Haluan kiittää myös Ylen suosittelutiimin kehittäjiä tuesta ja avusta Areenan suosittelujärjestelmän kanssa. He tarjosivat myös suosittelujärjestelmälle ajoympäristön tutkimuksen ajaksi. Kiitän myös apulaisprofessori Arto Klamia kaikista hyvistä neuvoista ja kommenteista, jotka auttoivat minua saattamaan työn hyvään lopputulokseen.

Helsingissä 14.2.2019

Tuukka Paukkunen

Sisältö

1 Johdanto

Ennen internetin olemassaoloa kuluttajat ovat hankkineet tuotteet useimmiten kivijalkakaupoista, joissa tuotteet ovat hyllyissä esillä. Fyysisessä kaupassa kauppiaan täytyy tehdä valintoja siitä, mitä tuotteita hän pitää kauppansa valikoimassa [9]. Kuluttajan valittavana ovat siis vain ne tuotteet, jotka kauppias on katsonut parhaaksi asettaa myytäväksi. Kauppiaan päätöksentekoa ohjaa esimerkiksi kauppakiinteistön hyllytilan määrä, joka rajaa myytävänä olevien tuotteiden määrää. Suurempi hyllytila edellyttää kiinteistöltä suurempaa pinta-alaa, joka taas tarkoittaa suurempia kustannuksia. Muita rajoittavia tekijöitä ovat tuotteiden logistiikka ja välivarastointi. Tähän liittyvä käsite on Pareton periaate (engl. Pareto principle), jonka mukaan pieni osa myytävistä tuotteista (esimerkiksi noin 20 %) aiheuttaa suuren osan myynnistä (esimerkiksi noin 80 %) [9].

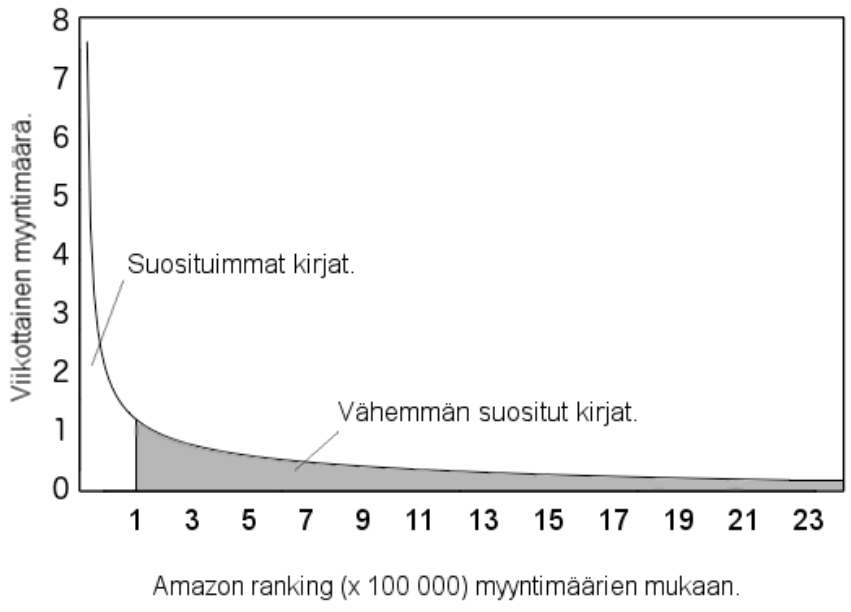
Internet on muuttanut tätä tilannetta. Kun kaupankäynti tai sisällön kuluttaminen tapahtuu internetin välityksellä, monet edellä mainituista rajoitteista esimerkiksi myyntikatalogin koon suhteen eivät enää päde. Fyysisten tuotteiden osalta keskitetty varastointi ja koordinoidut tuotteiden toimituskäytännöt mahdollistavat huomattavasti suuremman myyntikatalogin kuin kivijalkakaupoissa [11]. Virtuaalisten tuotteiden osalta varastointi on entistä yksinkertaisempaa, sillä varastointi edellyttää ainoastaan tuotteen taltioimista tietokantaan. Brynjolfsson, Smith ja Hu [10] vertailivat tuotevalikoimien kokoja tyypillisissä kivijalkakaupoissa ja Amazonin verkkokaupassa (taulukko 1) [10]. Esimerkiksi tarjolla olevien kirjojen määrä kivijalkakaupoissa oli 40 000 - 100 000, kun taas Amazonissa 2 300 000.

Kuvassa 1 nähdään kuvaaja kirjojen myyntimääristä Amazonin verkkokaupassa. Myytävät tuotteet, tässä tapauksessa kirjat, on sijoitettu kuvaajassa vaaka-akselille. Pystyakseli kuvaa kunkin tuotteen myyntimäärää. Kun tuotteet vaaka-akselilla jär-

Taulukko 1: Tuotevalikoiman vertailu verkkokaupan ja kivijalkakauppojen välillä [10].

Tuote	Amazon.com	Tyypillinen kivijalkakauppa
Kirjat	2 300 000	40 000 - 100 000
CD:t	250 000	5 000 - 1 500
DVD:t	18 000	500 - 1 500
Digitaaliset kamerat	213	36
Kannettavat MP3-soittimet	128	16
Skannerit	171	13

jestetään suosituimmuusjärjestykseen siten, että vasemmalla sijaitsevat suositut tuotteet ja oikealla vähemmän suositut, kuvaajasta tulee voimakkaasti laskeva käyrä. Mikäli kyse olisi fyysisessä kaupassa myytävistä tuotteista, tällöin kyseisen kaupan johto asettaisi johonkin kohtaan vaaka-akselille rajan, kuinka suosittu tuotteen täytyy vähintään olla, jotta se olisi järkevää laittaa myyntiin [11]. Fyysisessä kaupassa myytäväksi jäävät siis kuvaajassa tämän rajan vasemmalla puolella olevat tuotteet. Kuten kuvaajasta voidaan silmämääräisesti arvioida, vähemmän suosittujen kirjojen myynnin muodostama pinta-ala on varsin huomattava johtuen tuotteiden suuresta lukumäärästä. Ilmiötä kutsutaan nimellä ”pitkä häntä” (engl. long tail), joka kuvaa sitä, että pienemmän markkinasegmentin tuotteiden eli niin sanottujen ”niche”-tuotteiden kysyntä voi muodostaa ison osan myynnin kokonaismäärästä [3]. Brynjolfsson, Hu ja Simester [9] tarjoavat kaksi seikkaa pitkä häntä -ilmiön selitykseksi. Kuten taulukosta 1 nähdään, internetin myyntikanavat pystyvät tarjoamaan huomattavasti suuremman tuotevalikoiman kuin perinteiset kanavat. Toisekseen internet mahdollistaa kuluttajille kätevämmän ja halvemman tavan hankkia tietoa tuotteista, jolloin pienemmän markkinasegmentin tuotteiden kysyntä kasvaa.



Kuva 1: Long tail -ilmiö kirjojen myyntimäärissä.

Kun kaupankäynti ja sisältöjen tarjoaminen ja kuluttaminen on siirtynyt internetiin, ongelma ei ole enää se, ettei tuotetta ole tarjolla vaan se, että laajasta tuotevalikoimasta on vaikeaa on löytää haluttua tuotetta [11]. Yhtenä ratkaisuna tähän ongelmaan tarjotaan automaattista sisältöjen suosittelua [16]. Suosittelujärjestelmillä on selvä paikkansa tilanteissa, joissa tarjolla olevan informaation määrä ylittää laajalti yksilön kyvyn kartoittaa sitä [12]. Brynjolfsson, Hu ja Smith [11] toteavat artikkelis-

saan automaattisen sisältösuosittelun tasoittavan edellä mainittua Pareton periaatetta. Heidän mukaansa hakukoneiden ja suosittelujärjestelmien kaltaisten työkalujen käyttö liittyy pienemmän markkinasegmentin tuotteiden kasvaneeseen myyntiin.

Toisaalta Fleder ja Hosanagar [16] ovat tehneet tutkimuksessaan havaintoja, jotka haastavat edellä mainitut käsitykset suosittelualgoritmeista. Heidän mukaansa niillä saattaa olla kolmenlaisia vaikutuksia. Suosittelualgoritmit saattavat vaikuttaa siten, että suosittujen kohteiden suosio kasvaa entisestään. Toisekseen suosittelujärjestelmät saattavat kyllä kasvattaa monimuotoisuutta (engl. diversity) yksilötasolla, mutta toisaalta laskea monimuotoisuutta kokoomatasolla. Tällä tarkoitetaan sitä, että suosittelujärjestelmät voivat ohjata yksittäisiä käyttäjiä uusien tuotteiden pariin, mutta toisaalta ne saattavat myös ohjata samankaltaisia käyttäjiä samojen tuotteiden pariin. Kolmannen heidän havaintonsa mukaan suosittelujärjestelmän suunnittelulla on edellä mainituissa tuloksissa merkittävä osa. Suosittelujärjestelmien suunnittelulla voidaan siis vaikuttaa siihen, minkälaiset tuotteet saavat suosittelujärjestelmän avulla näkyvyyttä.

Edellisissä kappaleissa käsiteltiin fyysisten tuotteiden kaupankäyntiä verkkokaupoissa. Samat lainalaisuudet pätevät myös virtuaalisiin tuotteisiin. Tässä tutkielmassa keskitytään Ylen lastenohjelmiin. Finnpanelin tutkimuksen mukaan viikolla 39 vuonna 2018 50 katsotuimman ohjelman joukosta 12 ohjelmaa ovat lastenohjelmia [39]. Lastenohjelmiksi luetaan ne ohjelmat, jotka löytyvät Ylen Lasten Areenasta. Listan 50 ohjelman yhteenlaskettu käynnistysten lukumäärä on 1 831 000 kappaletta. 12 suosituimman lastenohjelman käynnistysten yhteenlaskettu määrä on 554 000 kappaletta, joka on noin 30 % kaikista käynnistyksistä. Tämän perusteella lapset muodostavat Yle Areenan käyttäjäkunnasta huomattavan osan. Tällöin myös lapset ovat edellä mainittujen, muun muassa sisällön määrää koskevien pulmien äärellä. Lasten Areenassa ei tällä hetkellä kuitenkaan ole käytössä automaattista sisältöjen suosittelua. Lasten suosittelua on tutkittu julkaistujen artikkelien lukumäärän perusteella toistaiseksi vähän. Vuonna 2017 järjestettiin kuitenkin ensimmäinen Kidrec-konferenssi lasten suosittelusta, joka oli hyvä avaus tutkimusta kaipaavalla alueella.

Tämän Pro gradu -tutkielman aiheena on selvittää, miten automaattista sisältöjen suosittelua voidaan tuottaa Lasten Areenaan. Tutkielmassa mitataan Yle Areenan olemassa olevan suosittelujärjestelmän soveltuvuutta Lasten Areenan suositteluun. Järjestelmästä esitetään tähän tarpeeseen myös muokattu versio, jossa mallin opetus tapahtuu vain lastenohjelmilla. Lisäksi tuotetaan muistipohjaiseen algoritmiin perustuva suosittelujärjestelmä, jonka mittaustuloksia verrataan Yle Areenan suosittelujärjestelmän tuloksiin. Tutkielmassa paneudutaan myös siihen, minkälaisia

lähdetietoja suosittelulle on tarjolla ja millä tavoin niitä on hyödyllistä käyttää.

Tutkielman luvussa 2 kuvataan suosittelualgoritmien käytön taustalla olevia motiiveja ja eri sidosryhmien tarpeita. Lisäksi syvennytään lapsille kohdistettujen sisältöjen suosittelun erityispiirteisiin. Luvussa 3 esitellään oleellisimmat suosittelualgoritmit sekä niiden matemaattiset taustat. Tämän jälkeen luvussa 4 esitellään tämän tutkielman tutkimuksen kohteina olevat suosittelujärjestelmät. Luvussa 5 käydään läpi tutkimuksen toteutustapa, siinä käytetyt lähdetiedot, mittaustavat, sekä esitellään tutkimuksen tulokset. Tuloksia ja niistä tehtäviä johtopäätöksiä pohditaan luvussa 6. Lopuksi tutkielmasta muodostetaan yhteenveto luvussa 7.

2 Suosittelualgoritmien käyttötarkoitukset ja periaatteet

Tässä luvussa syvennytään suosittelualgoritmien käyttötarkoituksiin ja niiden taustalla oleviin periaatteisiin.

2.1 Yleiset periaatteet

Paul Resnick ja Hal Varian [30] määrittelivät suosittelujärjestelmät vuonna 1997 artikkelissaan järjestelmiksi, joissa "ihmiset syöttävät suosituksiaan järjestelmään, joita järjestelmä kokoaa ja välittää eteenpäin sopiville vastaanottajille". Robin Burke [12] kuvaa suosittelujärjestelmiä näin: "Suosittelujärjestelmä on mikä tahansa järjestelmä joka tuottaa käyttäjälle yksilöllisiä suosituksia tai jolla on käyttäjää yksilöllisesti ohjaava rooli kohti kiinnostavia tai hyödyllisiä kohteita suuresta valikoimasta sisältökohteita". Resnick ja Varian viittaavat artikkelissaan etupäässä järjestelmiin, joissa käyttäjät eksplisiittisesti ilmaisevat mieltymyksensä tuotteista, jonka jälkeen järjestelmä välittää tietoa näistä käyttäjien mieltymyksistä muille käyttäjille. Jos tarkastellaan Yle Areenan nykyistä suosittelua, niin periaatteessa senkin suosittelu toimii, kuten Resnick ja Varian kuvaavat. Areenan suosittelussa lähdetietoina toimivat kuitenkin tiedot Areenan ohjelmien toistokerroista eksplisiittisen palautteen sijaan. Ihmiset siis syöttävät implisiittisesti suosituksiaan järjestelmään toistotietojen muodossa. Tämän perusteella järjestelmä tuottaa muille käyttäjille sisältösuosituksia. Suosittelujärjestelmien määritelmänä Burken määritelmä on kuitenkin selkeämpi.

Suosittelujärjestelmät ovat merkittävässä asemassa useissa internet-palveluissa [31]. Palvelut, kuten Amazon.com, Youtube, Netflix tai Spotify nojaavat toiminnassaan vahvasti suositteluun. Moni mediayhtiö kehittää suosittelujärjestelmiä osana toimintaansa. Esimerkiksi suoratoistopalvelu Netflix järjesti vuonna 2006 kilpailun, jossa se tarjosi miljoona dollaria sille, joka kykenee kehittämään merkittävästi paremman suosittelujärjestelmän kuin Netflixin silloinen järjestelmä oli [23]. Suosittelujärjestelmien kehitys on merkittävässä roolissa myös Yle Areenassa.

Ricci et al [31] luettelevat viisi asiaa, joita suosittelujärjestelmien käytöllä yleisesti tavoitellaan palveluntarjoajien näkökulmasta.

1. Myynnin edistäminen,
2. monipuolisuuden korostaminen myytävien tuotteiden valikoimassa,

3. käyttäjätyytyväisyyden lisääminen,
4. käyttäjäuskollisuuden lisääminen ja
5. ymmärryksen kasvattaminen käyttäjistä.

Sisältöjen käytön kasvattaminen on ilmeisin suosittelujärjestelmän käyttötarkoitus. Kaupallisissa palveluissa omistajat haluavat myydä jo myydyn tuotteen ohella toisenkin tuotteen, jota käyttäjä ei välttämättä löytäisi ilman suositusta. Ei-kaupallisetkin toimijat haluavat kasvattaa sisältöjensä kulutusta. Suosittelujärjestelmän avulla syntyvää parempaa ymmärrystä käyttäjistä halutaan käyttää hyväksi kunkin palvelun toimintaa kehitettäessä.

Herlocker et al [18] ovat tutkineet suosittelujärjestelmien käyttöä myös käyttäjien näkökulmasta. Palveluntarjoajien lisäksi käyttäjillä itsellään on luonnollisesti tavoitteita, joita suosittelujärjestelmien avulla halutaan täyttää. He ovat löytäneet sisältöjen suosittelun kontekstista 2 pääasiallista käyttäjien tavoitetta ja 8 muuta käyttäjien tavoitetta.

Kaksi pääasiallista tavoitetta ovat:

- **Kontekstisidonnaisessa annotoinnissa** (engl. annotation in context) olemassa olevassa kontekstissa, esimerkiksi jossakin Yle Areenan kategoriassa, käyttäjällä on tavoitteena löytää sisältöjoukosta ne sisältöobjektit, jotka kiinnostavat häntä. Kontekstisidonnaisessa annotoinnissa sisältöobjektien järjestystä ei muuteta, mutta käyttäjälle kerrotaan esimerkiksi ennuste, millä todennäköisyydellä kukin sisältöobjekti saattaisi häntä kiinnostaa.
- **Hyvien sisältöjen löytäminen** (engl. find good items): käyttäjälle tarjotaan henkilökohtaiseen kiinnostavuusjärjestykseen järjestetty lista sisältökohteita.

Sekä kahdeksan muuta tavoitetta ovat:

- **Kaikkien hyvien sisältöjen löytäminen** (engl. find all good items) on toisinaan olennaista. Esimerkiksi lakiasioita hoitavalle yhtiölle saattaa olla olennaista, että mikään mahdollisesti relevantti sisältöobjekti ei saa jäädä huomiotta.
- **Suosittelujatkumossa** (engl. recommend a sequence) käyttäjälle suositellaan yhden sisältöobjektin sijaan monta sisältöobjektia ikään kuin toistolistana.

- **”Olen vain katselemassa”** (engl. just browsing) -toiminnon tavoitteena on ohjata vain katselemassa olevaa käyttäjää hänelle sopivimpien sisältöjen pariin.
- **Uskottavan suosittelijan löytäminen** (engl. find credible recommender) on tärkeää niille käyttäjille, jotka eivät luota järjestelmän suosituksiin.
- **Käyttäjäprofiilin parantamisen** (engl. improve profile) tavoitteena on auttaa suosittelujärjestelmää tarjoamaan parempia suosituksia syöttämällä arvioita kuluttamistaan sisältöobjekteista.
- **Itseilmaisun** (engl. express self) tavoitteena on ilmaista itseään tuottamalla arvioita sisältöobjekteista. Suosittelujärjestelmä ei niinkään ole kiinnostuksen kohteena.
- **Toisten auttaminen** (engl. help others) voi olla myös motiivina käyttäjän kirjoittaessa arvioita sisältöobjekteista.
- **Toisten käyttäjien manipulointi** (engl. influence others) voi myös olla käyttäjän tavoitteena, mikäli käyttäjää syystä tai toisesta hyödyttää korostaa jonkin sisältöobjektin hyödyllisyyttä muille käyttäjille. Myös kaupalliset, kilpailuun liittyvät motiivit saattavat olla tällaisen toiminnan taustalla.

Hu, Koren ja Volinsky [19] korostavat sitä, että suositukset täytyisi perustella käyttäjälle. Käyttäjien luottamus suosittelujärjestelmää kohtaan kasvaa, kun käyttäjät näkevät, mitkä tekijät ovat vaikuttaneet suositusten syntyyn. Tällöin käyttäjät myös kykenevät ottamaan oikean näkökulman suosituksiin.

2.2 Lasten suosittelu ja sen tutkimustietoa

Kuten johdannossa todettiin, lapset median käyttäjinä ovat samojen haasteiden edessä kuin aikuisetkin median käyttäjät: tarjolla on valtavat määrät sisältöä, joista itselle relevantin sisällön löytäminen voi olla hankalaa. Suosittelujärjestelmät ovat siten samalla tavoin merkittäviä apuvälineitä mieluisan tai käyttäjälle hyödyllisen sisällön löytämisessä myös silloin, kun käyttäjinä ovat lapset.

Lapsen kehitysvaiheiden vaikutus suositteluun

Deldjoo et al [14] kuvaavat artikkelissaan sveitsiläisen psykologin Jean Piaget’n teorian lapsen kehitysvaiheista, jotka ovat:

1. Sensomotorinen vaihe (0 - 2 vuotta) (engl. Sensorimotor stage).
2. Esioperationaalinen vaihe (2 - 7 vuotta) (engl. Preoperational stage).
3. Konkreettisten operaatioiden vaihe (7 - 11 vuotta) (engl. Concrete operational stage).
4. Formaalien operaatioiden vaihe (11 vuodesta ylöspäin) (engl. Formal operational stage).

Sensomotorisessa vaiheessa lapsen toiminta ja ymmärrys rajoittuu voimakkaasti hänen omiin toimiinsa suhteessa hänen ympäristöönsä. Lapsen aistihavainnot ovat määräävä tekijä lapsen toiminnassa. Mediasta puhuttaessa vuorovaikutus sensomotorisessa vaiheessa olevan lapsen kanssa tulee tapahtua äänellä, kuvalla tai liikkuvilla kuvilla. Esioperationaalisessa vaiheessa lapselta voidaan odottaa jo jonkin verran vuorovaikutusta. Silti kaikkien vuorovaikutuskeinojen on edelleen nojattava ääneen tai kuviin.

Konkreettisten operaatioiden vaiheessa lapsi kykenee jo käyttämään sujuvasti esimerkiksi tietokoneen hiirtä tai näppäimistöä. Myös lapsen ajattelu on tässä vaiheessa kehittymässä. Kun lapsi saavuttaa formaalien operaatioiden vaiheen, hänen ajattelun voidaan sanoa olevan samalla tasolla kuin aikuisenkin. Toki maussa esimerkiksi median suhteen on iästä riippuen suuriakin eroavaisuuksia.

Huomionarvoista tässä on siis se, että kun puhutaan lapsista median käyttäjinä, puhutaan hyvin kirjavasta joukosta ihmisiä. Esimerkiksi 2-vuotias median käyttäjä on sekä tarpeiltaan että kyvyiltään täysin erilainen kuin 12-vuotias.

Haasteet lasten suosittelussa

Michael Ekstrand [15] on tutkinut lasten sisältösuosittelun haasteita. Merkittävä haaste on soveltuvan, julkisesti tarjolla olevan lähdetiedon puute. Lapsilta kerättävää dataa rajoittavat aivan oikeutetusti tarkat yksityisyyssäädökset. Esimerkiksi Yhdysvalloissa laki rajoittaa tietojen keräämistä alle 13-vuotiailta lapsilta. Tästä johtuen lapsia koskevia lähdetietokokoelmia ei juuri ole julkisesti saatavilla. Saatavilla olevat lähdetietokokoelmat rajoittuvat esimerkiksi sellaisiin, joista on tehty erilliset salassapitosopimukset. Tämän johdosta lapsille soveltuvan sisältösuosittelujärjestelmän kehittämisessä ei ole päässyt syntymään samankaltaista maailmanlaajuista kulttuuria tai yhteisöä kuin aikuisten sisältösuosittelujärjestelmissä.

Toisena haasteena Ekstrand mainitsee tutkimusten järjestämisen. Riittävän pitkäkestoisten tutkimusten läpivieminen aikuistenkin kanssa on haastavaa. Lapset tut-

kimuksen kohteina tuskin haluavat tai voivat osallistua erityisen pitkäkestoisiin tutkimuksiin. Lisäksi on huomioitava, että lapsilla iästä ja kehitysvaiheesta riippuen ei välttämättä ole tarvittavaa lukutaitoa tai tarvittavia kognitiivisia kykyjä ymmärtämään tutkimuksessa käsiteltäviä asioita.

Ekstrandin mukaan suurelle yleisölle tarkoitettujen suosittelujärjestelmien mallit on opetettu useimmiten lähdetietojoukoilla, jotka koostuvat implisiittisistä tiedoista, kuten klikkaukset, toistokerrat, ostoskerrat ja niin edespäin. Jo pelkästään aikuisten ollessa kyseessä päätelmien tekeminen näistä implisiittisistä lähdetiedoista on haastavaa verrattuna eksplisiittisen palautteen pohjalta toimivaan suositteluun. Lapsilla oppiminen ja kehittyminen on käynnissä koko ajan. He ovat keräämässä kykyjä suoriutua ympäröivän maailman tarjoamista haasteista. Ekstrandin mukaan lasten vielä kehittymässä oleva tiedonlukutaito johtaa siihen, että lasten implisiittiset lähdetiedot ovat hälyisämpiä kuin aikuisten. Lapsi saattaa esimerkiksi käynnistää saman ohjelman useita kertoja käyttöliittymän haasteiden vuoksi. Tämän lisäksi lasten antamat eksplisiittiset arviot sisältöobjekteista eivät välttämättä ole erityisen luotettavia.

Kun puhutaan sisältöjen suosittelusta aikuisille, tällöin sidosryhmänä on käytännössä vain henkilö itse, jolle sisältöjä suositellaan. Lasten suhteen tilanne on erilainen. Lapsella itsellään on toki tarpeita ja mielenkiinnon kohteita sisältöobjektien suhteen, mutta niitä on lisäksi lapsen ympärillä olevilla sidosryhmillä. Lapsen itsensä lisäksi lapsella on vanhemmat tai huoltaja, joilla luonnollisesti on näkemys siitä, minkälainen sisältö kyseiselle lapselle olisi sopivaa. Lisäksi lapsen koulu ja luokanopettaja voivat myös muodostaa oman sidosryhmän, jolla on lapsen kehittymisen suhteen näkemys sopivasta tai lapselle tarpeellisesta sisällöstä.

Markus Schedl ja Christine Bauer [33] olivat tutkineet lasten ja nuorten musiikin kuuntelutottumuksia verkossa. He tutkivat Last.fm -palvelusta peräisin olevia käyttötietoja 6-18-vuotiaiden lasten ja nuorten osalta. He tarkastelivat tämän käyttäjäryhmän musiikkigenrepreferenssejä sekä tutkivat, minkälaisia homogeenisiä ryhmiä tämän käyttäjäryhmän sisältä löytyy. He mittasivat yhteistoiminnallisen suosittelun suorituskykyä ja vertailivat tämän käyttäjäryhmän kaikkien käyttäjien tuloksia keskenään. He havaitsivat tuloksista, että tutkittavan ikäryhmän suositusten neliöllinen keskiarvovirhe (engl. Root Mean Square Error, RMSE) oli tuloksella 7,8 huomattavasti pienempi kuin koko väestön tuloksella 29,1. He tekivät tästä johtopäätöksen, että yhteistoiminnallinen suosittelu lapsille ja nuorille toimii erityisen hyvin johtuen siitä, että lapset ja nuoret muodostavat homogeenisempia ryhmiä musiikkimaun suhteen. He korostavat näiden tulosten olevan yhdenmukaisia kehityspsykologian tulosten kanssa, joiden mukaan musiikki on merkittävässä asemassa nuorten sosiaalisessa

kanssakäymisessä.

Haasteista huolimatta sisältöjen suosittelussa lapsille on paljon mahdollisuuksia. Ekstrand korostaa, että ottamalla rohkeasti useamman sidosryhmän näkemykset lasten suosittelussa huomioon, lopputulos voi olla suotuisa. Esimerkkinä Ekstrand mainitsee kirjojen suosittelujärjestelmän, joka voisi ottaa huomioon lapsen oman maun lisäksi myös vanhempien tai koulun näkemykset oppimisen ja kehittymisen suhteen ja siten suositella kirjoja, jotka täyttävät nämä kaikki tarpeet parhaiten. Käytännön esimerkkiä Ekstrand ei tässä yhteydessä kuitenkaan mainitse.

Yleisradion liiketoiminnan lähtökohdat lasten suosittelulle

Ylen lastenohjelmien periaatteet [38] toimivat Lasten Areenan suosittelulle luontevana perustana.

Lastenohjelmien periaatteet on luotu siitä lähtökohdasta, että lapsen asema on keskeinen. Ylen lastenohjelmien periaatteissa korostuu Ylen lastenohjelmien kasvatuksellinen tehtävä. Ohjelmiston tulee tukea lapsen kasvua ja positiivisen minäkuvan muodostumista. Ylen lastenohjelmien tulee myös auttaa lasta kohtaamaan ja hyväksymään tunteensa. Periaatteissa mainitaan erikseen myös se, ettei ohjelmisto sisällä viihteellistä tai perusteetonta väkivaltaa. Periaatteiden mukaisesti eri ikäisille lapsille tulee tarjota heille sopivaa ohjelmistoa. Kuten tässä luvussa aiemmin mainittiin, esimerkiksi 2-vuotias on hyvin erilainen median käyttäjä kuin vaikkapa 12-vuotias.

Suomen- ja ruotsinkieli, suomalainen kulttuuri sekä monikulttuurisuus ovat periaatteissa tärkeitä arvoja. Tämä ilmenee Ylen lastenohjelmistossa myös siten, että eri sukupuolta olevia, eri ikäisiä ja eri etnisen taustan omaavia ihmisiä pyritään näyttämään ohjelmissa mahdollisimman tasapuolisesti. Eräänä olennaisena piirteenä tässä on pyrkimys siihen, ettei kaikki ohjelmisto ole animaatioita vaan mukana on aitoja ihmisiä vuorovaikutuksessa keskenään. Ylellä on periaatteena tarjota lastenohjelmia pienen kielialueen lapsille heidän äidinkielellään. Periaatteiden mukaisesti ohjelmistossa tulee korostua suomalainen tapakulttuuri ja perinteet. Ylen lastenohjelmiston tulee auttaa lasta hahmottamaan paikkaansa maailmassa osana suomalaista yhteiskuntaa.

Ylen lastenohjelmisto on siis valittu tai tuotettu edellä mainittujen periaatteiden mukaisesti. Tällöin ohjelmiston voidaan katsoa jo noudattavan näitä periaatteita. Mikäli ohjelmisto ei olisi kokonaisuudessaan tasaveroisesti käyttäjien saatavilla, voidaan pohtia, toteutuvatko periaatteet käytännössä. Sen vuoksi periaatteilla on keskeinen asema myös Lasten Areenan suosittelussa. Suosittelutavan valinnassa keskeistä on se, että lasten suosikkiohjelmat, kuten esimerkiksi Ryhmä Hau, eivät saisi

saada suosituksissa tarpeettoman suurta painoarvoa. Sen sijaan sellaisten arvojen, kuten monipuolisuus, tasapuolisuus tai suomalainen kulttuuri, tulisi myös saada tarpeellinen näkyvyys suosituksissa. Tämän vuoksi lienee perusteltua, että Lasten Areenan suosittelun keskeisenä tehtävänä on tarjota kestosuosikkien lisäksi vaihtoehtoja, eli jotain sellaista, mitä lapsi ei ehkä itsekseen keksisi etsiä Areenan ohjelmalistausten seasta.

Muun muassa näitä asioita käsiteltiin Lasten Areenan suosittelua käsittelevässä työpajassa elokuussa 2018 Tampereella, jossa hahmoteltiin Lasten Areenan suosittelun taustalla olevia arvoja ja periaatteita. Työpajassa pohdittiin sopivaa tasapainoa, jossa lapselle toisaalta suositellaan mieluista sisältöä, mutta toisaalta ei uhrata Yleisradion lastenohjelmien edellä kuvattuja arvoja. Tavoitteena on, että suosittelu tukee lastenohjelmien periaatteita.

3 Katsaus suosittelualgoritmeihin

Seuraavaksi käydään läpi joukko yleisimpiä suosittelumenetelmiä.

Tässä tutkielmassa käytetään termejä ja kirjainlyhenteitä soveltaen ISO-standardin 80000-2 [20] määritelmiä. Englanninkielisissä teksteissä suosittelualgoritmien yhteydessä käytetään käsitteitä ”users” ja ”items”. Tässä tutkielmassa nämä suomennetaan sanoilla ”käyttäjä” ja ”sisältöobjekti”. Koska aiheena on Lasten Areenan suosittelu, sanan ”sisältöobjekti” sijaan käytetään myös sanaa ”ohjelma”.

Tietoa, joka voidaan koostaa matriisimuotoon, kuvataan suurella lihavoidulla ja kursivoidulla kirjaimella, esimerkiksi $\boldsymbol{A}$. Matriisin rivejä ja sarakkeita kuvataan alaindekseillä. Esimerkiksi $\boldsymbol{A}_{xy}$ kuvastaa matriisin $\boldsymbol{A}$ rivin x sarakkeen y arvoa. Tensoreita eli moniulotteisia tietorakenteita merkitään samalla tavalla, mutta alaindeksejä on tällöin useampia. Vektorit merkitään pienellä lihavoidulla ja kursivoidulla kirjaimella, esimerkiksi $\boldsymbol{q}$. Mikäli jostain muuttujasta halutaan antaa tarkempi kuvaus, esimerkiksi jos vektori $\boldsymbol{q}$ on ohjelmaa i kuvaava vektori, tällöin käytetään alaindeksiä $\boldsymbol{q}_i$. Mikäli ilmenee sekaannuksen mahdollisuus vektorin $\boldsymbol{q}$ i:nteen alkioon, tämä selvennetään tekstissä.

Matriisi $\boldsymbol{Y}$ tarkoittaa lähdetietojen joukkoa. Matriisin alkio $\boldsymbol{Y}_{ui}$ tarkoittaa sitä, että rivillä u sarakkeessa i on käyttäjän u arvio ohjelmasta i. Matriisin $\boldsymbol{Y}$ sisältämät arviot voivat olla joko implisiittisiä tai eksplisiittisiä. Käyttäjien kokonaismäärää kuvataan kirjaimella n ja ohjelmien kokonaismäärää kuvataan kirjaimella m. $\boldsymbol{Y}$ on siis $n \times m$ matriisi.

Ennuste tai prediktio $\boldsymbol{P}_{ui}$ on suosittelualgoritmin muodostama arvio siitä, miten käyttäjä u saattaisi arvioida hänelle entuudestaan tuntemattoman ohjelman i. Kirjain P on valittu englanninkielen sanan *prediction* mukaan. Ennusteista voidaan koostaa käyttäjäkohtainen vektori tai matriisin rivi $\boldsymbol{P}_u$, joka sisältää ennusteet kaikkien ohjelmien i arvioista käyttäjälle u. Jos jossain algoritmissa vertaillaan kahta ohjelmaa tai käyttäjää keskenään, niin tällöin vertailuparin osapuolia merkitään ohjelmien suhteen i ja i', ja käyttäjien suhteen u ja u'.

Suosittelujärjestelmätyyppejä

Teknisesti yksinkertaisimmillaan suosittelua tuotetaan siten, että verkkopalvelun toimitus tuottaa suosituksia verkkopalvelun sisällöstä käyttäjille käsityönä. Esimerkiksi Yle Areenassa osa ohjelmasuosituksista poimitaan automaattisen järjestelmän lisäksi käsin [37]. Automaattisesti luotavaa, mutta varsinaiseen suosittelujärjestelmään verrattuna hieman yksinkertaisempaa suosittelua ovat erilaiset automaattises-

ti tuotettavat listat, esimeriksi ”Suosituimmat sisällöt”.

Yleisesti ottaen suosittelujärjestelmät koostuvat kolmesta käsitteestä: 1) taustatieto, joka järjestelmään on taltioitu ennen kuin suositteluprosessi alkaa, 2) käyttäjän tarjoama syöte, jota käyttäjän täytyy tarjota järjestelmälle, jotta suositteluja voidaan tarjota, ja 3) algoritmi, joka yhdistää taustatiedot ja käyttäjän tarjoaman tiedon tarjotakseen suosituksia [12]. Esimerkiksi Yle Areenassa taustatietoina toimivat ohjelmat ja niiden metatiedot. Käyttäjien syötteenä toimii tieto siitä, minkälaisen osuuden käyttäjä on katsonut tai kuunnellut ohjelmasta. Algoritmi on luonnollisesti käytössä oleva suosittelualgoritmi.

Robin Burke jakaa suosittelumenetelmät artikkelissaan seuraaviin kokonaisuuksiin [12]:

- Yhteistoiminnallinen suosittelu (engl. *Collaborative recommendation*),
- sisältöön perustuva suosittelu (engl. *Content-based recommendation*),
- väestötieteellinen suosittelu (engl. *Demographic recommendation*),
- hyötyyn perustuva suosittelu (engl. *Utility-based recommendation*) ja
- käyttäjätietoon perustuva suosittelu (engl. *Knowledge-based recommendation*).

Näistä suosittelutavoista tässä luvussa syvennytään etupäässä yhteistoiminnalliseen suositteluun. Muita suosittelutapoja käsitellään yleisluontoisemmin.

3.1 Yhteistoiminnallinen suosittelu

Yhteistoiminnalliset suosittelumenetelmät (engl. collaborative filtering tai collaborative recommendation) tuottavat käyttäjäkohtaisia suosituksia tarjolla olevista sisältöobjekteista [31]. Yhteistoiminnallisen suosittelun tavoitteena on ennustaa tietyn sisältöobjektin hyödyllisyys tietylle käyttäjälle [8]. Lähdetietoina toimivat käyttäjien arviot sisältökohteista. Yhteistoiminnalliset suosittelumenetelmät siis kokoavat arvioita sisältöobjekteista, tunnistavat samankaltaisuuksia käyttäjien välillä heidän antamiensa arvioiden perusteella ja muodostavat uusia suosituksia käyttäjien mieltymyksiä vertailemalla [12].

Yhteistoiminnallisessa suosittelussa käyttäjien arviot sisältöobjekteista voivat olla eksplisiittisiä (engl. explicit votes) tai implisiittisiä (engl. implicit votes) [8]. Eksplisiittisessä arvioinnissa käyttäjä antaa arvion sisältöobjektista jollain asteikolla,

esimerkiksi yhdestä viiteen tähteä. Eksplisiittinen arviointi saattaa myös yksinkertaisesti olla tieto siitä, onko käyttäjä pitänyt sisältöobjektista (”peukku ylös”) vai ei (”peukku alas”). Implisiittisissä arvioissa sen sijaan on kyse siitä, että käyttäjän kiinnostusta sisältöobjekteja kohtaan pyritään tulkitsemaan hänen käytöksensä perusteella. Implisiittisessä arvioinnissa arviot voivat perustua esimerkiksi käyttäjän selailukäyttäytymisestä kertyneeseen tietoon. Implisiittinen arvio voi muodostua myös sen perusteella, mikäli käyttäjä on kuluttanut sisältökohdetta. Tällöin voidaan olettaa, että sisältökohde kiinnosti tai miellytti käyttäjää ainakin jollain tasolla [28]. Eksplisiittisen arvioinnin tarjoamat tiedot ovat implisiittistä arviointia tarkempia, mutta eksplisiittisten arvioiden keräämiseen liittyy kustannuksia. Implisiittistä sisällön arviointia käydään tarkemmin läpi luvussa 3.1.4.

Yhteistoiminnallinen suosittelu jakautuu kahteen luokkaan: muistipohjaiset algoritmit (engl. memory-based algorithms) ja mallipohjaiset algoritmit (engl. model-based algorithms) [8]. Muistipohjaiset algoritmit tekevät ennusteita käyttäjien mieltymyksistä käyttämällä hyväksi kaikkia tietolähteen tietoja kerralla. Mallipohjaiset algoritmit sen sijaan oppivat tietolähteen pohjalta mallin, jota käytetään ennusteiden tekemiseen.

Yhteistoiminnallisessa suosittelussa tietolähteenä ovat käyttäjäkohtaiset vektorit, joka sisältää käyttäjän arviot kustakin arvioimastaan sisältöobjektista [12]. Nämä vektorit voidaan koostaa matriisiksi $\boldsymbol{Y}$, jossa on rivi kullekin käyttäjälle u ja sarake kullekin ohjelmalle i. Mikäli käyttäjä u on arvioinut ohjelman i, tällöin arvio taltioidaan matriisiin $\boldsymbol{Y}$ riville u sarakkeeseen i. Tyypillisessä skenaariossa, esimerkiksi Yle Areenassa, suurin osa matriisista $\boldsymbol{Y}$ on täyttämättä, sillä on todennäköistä, että käyttäjät ovat arvioineet joko implisiittisesti tai eksplisiittisesti vain pienen osan kyseisen palvelun sisältöobjekteista.

3.1.1 Muistipohjaiset algoritmit

Muistipohjaisissa suosittelualgoritmeissa (engl. memory-based algorithms) ennuste $\boldsymbol{P}_{ui}$ käyttäjän u arviosta ohjelmasta i voidaan laskea kahdella tavalla: käyttäjäpohjaisesti (engl. user-based recommendation) tai sisältöobjektipohjaisesti (item-based recommendation) [19, 12, 8]. Hu, Koren ja Volinsky [19] mainitsevat artikkelissaan, että muistipohjaiset yhteistoiminnalliset suosittelualgoritmit olivat alkuperäisessä muodossaan käyttäjäpohjaisia. Myöhemmin muistipohjaisissa algoritmeissa suosiota saavutti myös sisältöobjektien samankaltaisuuksiin perustuva menetelmä.

Käyttäjäpohjaisessa suosittelussa ennuste käyttäjän u arviosta ohjelmasta i lasketaan tekemällä painotettu summa muiden käyttäjien u' antamista arvioista ohjel-

malle i [8]. Kunkin käyttäjän u' arvio painotetaan käyttäjien u ja u' samankaltaisuudella. Tällöin käyttäjäpohjaisessa suosittelussa käyttäjän u kanssa samankaltaisten käyttäjien arviot saavat suuremman painoarvon. Sen sijaan sisältöobjektipohjaisessa suosittelussa ennuste käyttäjän u kiinnostuksesta ohjelmalle i muodostetaan laskemalla summa käyttäjän u toistamien muiden ohjelmien i' arvioista [32]. Kukin näistä arvioista painotetaan sen mukaan, kuinka samankaltaisia ohjelmat i ja i' ovat.

Käyttäjäpohjainen suosittelu

Käyttäjäpohjaisissa suosittelualgoritmeissa (engl. user-based algorithms) ennuste $\boldsymbol{P}_{ui}$ käyttäjälle u ohjelmasta i lasketaan laskemalla painotettu summa muiden käyttäjien arvioista ohjelmalle i [8]. Ennuste lasketaan kaavalla

$$\boldsymbol{P}_{ui} = \overline{Y}_u + K_u \sum_{u'} \boldsymbol{W}_{uu'}(\boldsymbol{Y}_{u'i} - \overline{\boldsymbol{Y}}_{u'}),$$

missä $\overline{Y}_u$ on käyttäjän u arvioimien ohjelmien arvioiden keskiarvo. Kukin painotus $\boldsymbol{W}_{uu'}$ kuvastaa käyttäjien u ja u' samankaltaisuutta. K_u on normalisointiarvo, jonka avulla painotusten itseisarvot summautuvat 1:een.

Intuitiivisesti käyttäjäpohjaista suosittelua voisi kuvata siten, että ennuste käyttäjän u arviosta ohjelmasta i lasketaan muiden käyttäjien arvioista samalle ohjelmalle i, painottaen kunkin käyttäjän arviota sen mukaan, kuinka samankaltaisia käyttäjä u ja toinen käyttäjä u' ovat.

Käyttäjien samankaltaisuutta mitataan esimerkiksi Pearson-korrelaatiokertoimella [8]. Käyttäjien u ja u' samankaltaisuutta kuvaa tällöin

$$\boldsymbol{W}_{uu'} = \frac{\sum_i (\boldsymbol{Y}_{ui} - \overline{Y}_u)(\boldsymbol{Y}_{u'i} - \overline{Y}_{u'})}{\sqrt{\sum_i (\boldsymbol{Y}_{ui} - \overline{Y}_u)^2 \sum_i (\boldsymbol{Y}_{u'i} - \overline{Y}_{u'})^2}},$$

jossa $\overline{Y}_u$ kuvastaa keskiarvoa käyttäjän u arvioimien ohjelmien arvioista ja $\boldsymbol{Y}_{ui}$ on käyttäjän u antama arvio ohjelmalle i. Arvot $\boldsymbol{Y}_{u'i}$ ja $\overline{Y}_{u'}$ noudattavat samaa logiikkaa. Summat ohjelmien i yli koskevat vain niitä ohjelmia, jotka sekä käyttäjä u että käyttäjä u' ovat arvioineet. Tämä rajoite on olemassa siitä syystä, että mikäli summassa käytäisiin läpi kaikki ohjelmat, tällöin esimerkiksi käyttäjän u katsomattoman ohjelman i kohdalla laskutoimituksesta $\boldsymbol{Y}_{ui} - \overline{Y}_u$ seuraisi se, että tiedon puuttumista kuvaava matriisin arvo $\boldsymbol{Y}_{ui} = 0$ tulkittaisiin virheellisesti negatiiviseksi arvioksi kyseisestä ohjelmasta.

Pearson-korrelaatiokerrointa laskettaessa käydään läpi kaikki ohjelmat i, joten yhden

käyttäjäparin korrelaation laskeminen sujuu lineaarisessa ajassa $\mathcal{O}(m)$, jossa m on ohjelmien lukumäärä.

Kun tämän jälkeen lasketaan ennuste $\boldsymbol{P}_{ui}$ käyttäjälle u ohjelmasta i, on oltava laskettuna käyttäjän u ja kaikkien muiden käyttäjien u' korrelaatiot. Käyttäjien lukumäärää merkitään kirjaimella n. Tällöin näiden korrelaatioiden laskemiseen tarvittava aikavaativuus on $\mathcal{O}(nm)$, joka on myös ennusteen $\boldsymbol{P}_{ui}$ laskemisen aikavaativuus.

Ohjelmapohjainen suosittelu

Ohjelmapohjaiset, tai yleisemmin sisältöobjektipohjaiset suosittelualgoritmit (engl. item-based algorithms) pyrkivät löytämään suhteita eri ohjelmien välillä ja laskevat suositteluja käyttäjälle näiden suhteiden perusteella [32]. Intuitiivisesti asian voisi ilmaista siten, että sisältöpohjaiset suosittelualgoritmit etsivät käyttäjälle suositeltavaksi samankaltaisia ohjelmia, joista käyttäjä on aiemmin pitänyt.

Tässä tutkielmassa käytetään tästä suosittelutavasta pääsääntöisesti nimitystä ohjelmapohjainen suosittelu, sillä tutkielma käsittelee nimenomaan medioiden suosittelualgoritmeja. Silloin, kun tässä tutkielmassa puhutaan tästä suosittelualgoritmista (engl. item-based algorithm) yleisemmin, puhutaan sisältöobjektipohjaisesta suosittelusta.

Ohjelmapohjaiset suosittelualgoritmit muodostavat ennusteen käyttäjän u arviosta ohjelmasta i laskemalla yhteen arviot muista käyttäjän u katsomista tai kuuntelemista ohjelmista i' ja kertomalla arviot ohjelmien i ja i' samankaltaisuudella [32, 19]. Tulos skaalataan halutulle arvoalueelle jakamalla tulos samankaltaisuuksien $\boldsymbol{W}$ itseisarvojen summalla, eli

$$\boldsymbol{P}_{ui} = \frac{\sum_{i'} \boldsymbol{W}_{ii'} \boldsymbol{Y}_{ui'}}{\sum_{i'} |\boldsymbol{W}_{ii'}|}.$$

Summissa $\sum_{i'}$ käydään läpi vain ne ohjelmat i', joille käyttäjä u on antanut arvionsa, joko eksplisiittisesti tai implisiittisesti.

Sarwar et al [32] luettelevat kaksi tapaa ohjelmien samankaltaisuuksien $\boldsymbol{W}$ laskemiseen: kosinipohjainen samankaltaisuus ja korrelaatiopohjainen samankaltaisuus.

Ohjelmien i ja i' saamat arviot muodostavat kaksi vektoria tai matriisin saraketta $\boldsymbol{Y}_i$ ja $\boldsymbol{Y}_{i'}$. Kosinipohjaisen samankaltaisuuden laskemisessa samankaltaisuus mitataan laskemalla näiden kahden vektorin välisen kulman kosini, eli

$$\boldsymbol{W}_{ii'} = cos(\boldsymbol{Y}_i, \boldsymbol{Y}_{i'}) = \frac{\boldsymbol{Y}_i \cdot \boldsymbol{Y}_{i'}}{\|\boldsymbol{Y}_i\|_2 \|\boldsymbol{Y}_{i'}\|_2},$$

jossa "$\cdot$" tarkoittaa vektorien sisätuloa.

Ohjelmapohjaisessa suosittelussa lasketaan ohjelmien samankaltaisuuksia mittaavat Pearson-korrelaatiokertoimet käyttäjäpohjaisen suosittelun tapaan. Ohjelmien i ja i' välinen korrelaatio

$$\boldsymbol{W}_{ii'} = \frac{\sum_u (\boldsymbol{Y}_{ui} - \overline{Y}_i)(\boldsymbol{Y}_{ui'} - \overline{Y}_{i'})}{\sqrt{\sum_u (\boldsymbol{Y}_{ui} - \overline{Y}_i)^2 \sum_u (\boldsymbol{Y}_{ui'} - \overline{Y}_{i'})^2}}.$$

Laskutoimituksessa summa $\sum_u$ koskee vain niitä ohjelma-arvioita, joissa käyttäjä u on arvioinut sekä ohjelman i että ohjelman i'.

Mikäli myös ohjelmapohjaisessa suosittelussa samankaltaisuudet lasketaan Pearson-korrelaatiokertoimen avulla, tällöin Pearson-korrelaatiokertoimen laskemisessa käydään läpi kaikki käyttäjät u, joten yhden ohjelmaparin korrelaation laskeminen sujuu lineaarisessa ajassa $\mathcal{O}(n)$, jossa n on käyttäjien lukumäärä.

Kun tämän jälkeen lasketaan ennuste $\boldsymbol{P}_{ui}$ käyttäjälle u ohjelmasta i, on oltava laskettuna ohjelman i ja kaikkien muiden ohjelmien i' korrelaatiot. Tällöin näiden korrelaatioiden laskemisen aikavaativuus on $\mathcal{O}(nm)$, joka on myös ennusteen $\boldsymbol{P}_{ui}$ laskemisen aikavaativuus. Muistipohjaisista algoritmeista käyttäjäpohjainen ja ohjelmapohjainen suosittelu eivät siis asymptoottisen aikavaativuuden näkökulmasta eroa toisistaan.

Käytännössä kuitenkin eroavaisuuksia löytyy. Käyttäjäpohjaisessa suosittelussa ennusteen arvo käyttäjälle u ohjelmalle i lasketaan muiden käyttäjien arvioiden perusteella, eli siinä käydään läpi muiden käyttäjien arviot ohjelmalle i. Ohjelmapohjaisessa suosittelussa sen sijaan käydään läpi saman käyttäjän arvioimat muut ohjelmat. On mahdollista, että ohjelman i keräämä katsojakunta on suurempi kuin yksittäisen käyttäjän u katsomien ohjelmien määrä. Sen vuoksi käyttäjäpohjaisessa suosittelussa käytännössä läpikäytävää materiaalia saattaa olla enemmän kuin ohjelmapohjaisessa suosittelussa.

3.1.2 Mallipohjaiset algoritmit

Ricci et al [31] luettelevat mallipohjaisiksi suosittelualgoritmeiksi (engl. model-based algorithms) muun muassa bayesilaisen klusteroinnin (engl. Bayesian Clustering), piilevän semanttisen analyysin (engl. Latent Semantic Analysis), piilevän Dirichlet-allokoinnin (engl. Latent Dirichlet Allocation), maksimientropian (engl. Maximum Entropy), Bolzmannin koneet, tukivektorikoneet (engl. Support Vector Machines) ja pääakselihajotelmat (engl. Singular Value Decomposition). Koska Yle Areenan

suosittelu perustuu pääakselihajotelmiin, keskitytään tässä luvussa siihen.

Pääakselihajotelmiin eli matriisihajotelmiin (engl. matrix factorization) perustuvat mallipohjaiset suosittelualgoritmit kuvaavat sekä käyttäjät että ohjelmat piilevien tekijöiden (engl. latent factor) avulla [23]. Nämä piilevät tekijät kuvataan f-ulotteisina vektoreina siten, että yksi vektori kuvaa yhtä käyttäjää tai yhtä ohjelmaa. Käyttäjä- ja ohjelmakohtaiset ennusteet saadaan laskemalla näiden vektoreiden sisätulo. Kullakin käyttäjällä u on vektori $\boldsymbol{p}_u \in \mathbb{R}^f$ ja kullakin ohjelmalla i on vektori $\boldsymbol{q}_i \in \mathbb{R}^f$. Kunkin vektorin $\boldsymbol{q}_i$ alkiot kuvastavat, missä määrin ohjelma i on kunkin piilevän tekijän kaltainen. Kunkin vektorin $\boldsymbol{p}_u$ alkiot kuvastavat, missä määrin käyttäjä u on kiinnostunut kunkin piilevän tekijän kaltaisesta sisällöstä. Piilevien tekijöiden voisi ajatella kuvastavan ohjelmien eri ominaisuuksia, ikään kuin kategorioita. Esimerkiksi: ”ohjelma on paljolti *tällainen*, mutta vain vähän *tuollainen*”, tai käyttäjistä puhuttaessa: ”käyttäjä pitää paljon *tällaisista* ohjelmista, muttei niinkään *tuollaisista* ohjelmista”. Piilevistä tekijöistä käytetään tässä tutkielmassa pääosin nimitystä komponentti.

Sisätulo $\boldsymbol{q}_i{}^\intercal \boldsymbol{p}_u$ kuvastaa käyttäjän u ja ohjelman i vuorovaikutusta, eli käyttäjän u kiinnostuksen määrää ohjelmaa i kohtaan. Tämä toimii ennusteena sille, miten käyttäjä u arvioisi ohjelman i, jos olisi katsonut tai kuunnellut sen. Tällöin ennuste

$$\boldsymbol{P}_{ui} = \boldsymbol{q}_i{}^\intercal \boldsymbol{p}_u.$$

Piilevät tekijät opitaan koneoppimisen avulla minimoimalla neliöllinen virhe (engl. squared error) suhteessa tunnettuihin ohjelma-arvioihin $\boldsymbol{Y}$. Virhefunktio on

$$\mathcal{L}(\boldsymbol{q}, \boldsymbol{p}) = \sum_{(ui)\in K} (\boldsymbol{Y}_{ui} - \boldsymbol{q}_i{}^\intercal \boldsymbol{p}_u)^2 + \lambda(\|\boldsymbol{q}_i\|^2 + \|\boldsymbol{p}_u\|^2), \tag{3.1}$$

jossa K on joukko käyttäjän ja ohjelman pareja (u, i), joissa käyttäjä u on arvioinut ohjelman i, tai implisiittisessä mallissa katsonut tai kuunnellut sen. Muuttujat $\boldsymbol{q}_i$ ja $\boldsymbol{p}_u$ ovat siis f-ulotteisia vektoreita, jossa f on etukäteen valittu halutuksi suuruudeksi.

Kuten edellä mainittiin, järjestelmä sovittaa mallin siten, että mallin ennustamien arvojen ja tunnettujen ohjelma-arvioiden välinen virhe olisi mahdollisimman pieni. Malli ei saa kuitenkaan tulla ylisovittuneeksi siten, että tunnetut ohjelma-arviot tavoitetaan mallissa hyvin täsmällisesti, mutta vielä tuntemattomissa ohjelma-arvioissa virhe olisi suuri. Tavoitteena siis on tehdä mallista yleinen siten, että tilaa jää tuleville, vielä tuntemattomille ohjelma-arvioille. Vakio λ säätää säännöllistämisen (engl. regularization) määrää. Vakion λ arvo määritellään yleisesti ristiinvalidoimisen avul-

la [23]. Ristiinvalidoinnissa mallin opetusmateriaali pilkotaan useaan osaan, joista eri osiot vuorollaan toimivat opetus- tai testimateraalina.

Optimointialgoritmit

Koren, Bell ja Volinsky [23] tarjoavat mallin sovittamisessa käytettäväksi optimointialgoritmiksi kahta vaihtoehtoa: stokastista gradienttimenetelmää (engl. Stochastic Gradient Descent, SGD) ja vuorottelevaa neliövirheen minimointia (engl. Alternating Least Squares, ALS).

SGD:ssä algoritmi käy läpi opetusmateriaalin ohjelma-arvioita iteroiden poimimalla otoksen opetusmateriaalista kullakin iteraatiokerralla. Järjestelmä ennustaa kullekin käyttäjälle u ja kullekin ohjelmalle i ennusteen $\boldsymbol{P}_{ui} = \boldsymbol{q}_i{}^\intercal \boldsymbol{p}_u$ ja laskee ennusteen virheen

$$\mathcal{L}(\boldsymbol{q}, \boldsymbol{p}) = (\boldsymbol{Y}_{ui} - \boldsymbol{q}_i{}^\intercal \boldsymbol{p}_u)^2 + \lambda(\|\boldsymbol{q}_i\|^2 + \|\boldsymbol{p}_u\|^2).$$

Kuten Takács, Pilászy ja Németh [35] artikkelissaan tarkemmin kuvaavat, neliövirhefunktion 3.1 gradientti on

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \boldsymbol{q}_i} &= -2(\boldsymbol{Y}_{ui} - \boldsymbol{q}_i{}^\intercal \boldsymbol{p}_u)\boldsymbol{p}_u + 2\lambda \boldsymbol{q}_i, \\ \frac{\partial \mathcal{L}}{\partial \boldsymbol{p}_u} &= -2(\boldsymbol{Y}_{ui} - \boldsymbol{q}_i{}^\intercal \boldsymbol{p}_u)\boldsymbol{q}_i + 2\lambda \boldsymbol{p}_u.\end{aligned}$$

Tämän jälkeen vektoreita $\boldsymbol{q}$ ja $\boldsymbol{p}$ päivitetään gradientin vastakkaiseen suuntaan

$$\begin{aligned}\boldsymbol{q}_i &:= \boldsymbol{q}_i + \gamma(2(\boldsymbol{Y}_{ui} - \boldsymbol{q}_i{}^\intercal \boldsymbol{p}_u)\boldsymbol{p}_u - 2\lambda \boldsymbol{q}_i), \\ \boldsymbol{p}_u &:= \boldsymbol{p}_u + \gamma(2(\boldsymbol{Y}_{ui} - \boldsymbol{q}_i{}^\intercal \boldsymbol{p}_u)\boldsymbol{q}_i - 2\lambda \boldsymbol{p}_u),\end{aligned}$$

jossa merkintä := tarkoittaa sitä, että kaavan oikealla puolella olevasta osiosta muodostetaan vektoreiden $\boldsymbol{q}$ ja $\boldsymbol{p}$ uudet arvot. Vakio γ on säädettävä askelpituus, joka kertoo, kuinka paljon vektoreita päivitetään.

Koren, Bell ja Volinsky ovat sisällyttäneet yllä olevasta funktiosta kertoimet 2 vakioon γ. Funktioon lisätään vakinaistaminen kertoimen λ verran, jolloin saadaan funktiot, joilla uudet vektorien $\boldsymbol{q}$ ja $\boldsymbol{p}$ arvot ovat

$$\boldsymbol{q}_i := \boldsymbol{q}_i + \gamma((\boldsymbol{Y}_{ui} - \boldsymbol{q}_i{}^\intercal \boldsymbol{p}_u)\boldsymbol{p}_u - \lambda \boldsymbol{q}_i),$$
$$\boldsymbol{p}_u := \boldsymbol{p}_u + \gamma((\boldsymbol{Y}_{ui} - \boldsymbol{q}_i{}^\intercal \boldsymbol{p}_u)\boldsymbol{q}_i - \lambda \boldsymbol{p}_u).$$

Yleisemmin gradienttimenetelmän periaate on seuraava. Oletetaan, että $\boldsymbol{a}$ toimii funktion parametrijoukkona. Oletetaan myös, että funktio F on konveksi ja että F mittaa virhettä toiminnassa, johon parametrijoukko $\boldsymbol{a}$ liittyy. Parametrijoukon $\boldsymbol{a}$ uudet arvot $\boldsymbol{a}_{n+1}$ saadaan, kun

$$\mathbf{a}_{n+1} = \mathbf{a}_n - \gamma \nabla F(\mathbf{a}_n),$$

jossa γ on vakio, jolla säädetään gradientin laskeutumisen nopeutta. ∇ kuvaa funktion F osittaisderivointia parametrijoukon $\boldsymbol{a}$ kunkin elementin mukaan. Kun osittaisderivoituun funktioon asetetaan arvoksi edellinen parametrijoukon $\boldsymbol{a}$ arvo, tuloksena saadaan negatiivinen luku, mikäli parametrin $\boldsymbol{a}_i$ arvoa täytyy kasvattaa, tai positiivinen luku, mikäli parametrin $\boldsymbol{a}_i$ arvoa täytyy vähentää.

SGD:ssä löydetään virhefunktion globaali minimi luotettavasti vain, jos virhefunktio on konveksi [7]. Koska funktiossa 3.1 on kuitenkin kaksi tuntematonta ($\boldsymbol{q}$ ja $\boldsymbol{p}$), funktio ei ole konveksi [23]. Tästä seuraa se, että optimiratkaisun löytyminen SGD:n avulla ei ole varmaa. Käytännössä menetelmää kuitenkin sovelletaan laajalti ja se tyypillisesti toimii hyvin.

Vuorotteleva neliövirheen minimointi

Vuorotteleva neliövirheen minimointi (engl. Alternating Least Squares, ALS), on vaihtoehtoinen tapa optimoida vektoreiden $\boldsymbol{q}$ ja $\boldsymbol{p}$ piileviä tekijöitä [23].

ALS:ssä asetetaan vuorotellen $\boldsymbol{q}$ ja $\boldsymbol{p}$ vakioiksi. Kun kaikki vektorit $\boldsymbol{q}_u$ ovat vakioita, järjestelmä laskee vektorit $\boldsymbol{q}_i$ minimoimalla neliövirheen. Samalla tavoin toimitaan vektorien $\boldsymbol{q}_i$ kanssa. Tätä vuorottelua jatketaan, kunnes funktion 3.1 arvo konvergoituu, eli ei saavuta enää parempia arvoja.

Kuten edellä mainittiin, funktio 3.1 ei ole konveksi. Jos kuitenkin joko $\boldsymbol{q}$ tai $\boldsymbol{p}$ kiinnitetään vakioksi, tällöin funktiosta tulee konveksi, jolloin optimointiongelma on helppo ratkaista [6].

Yunhong Zhou et al [40] tekivät ALS-algoritmille hajautetun toteutuksen hyvin tuloksin. ALS pystyttiin hajauttamaan, koska siinä kukin vektoreista $\boldsymbol{q}_i$ on riippumaton toisista vektoreista $\boldsymbol{q}_{i'}$, ja kukin vektoreista $\boldsymbol{p}_u$ on riippumaton toisista vektoreis-

ta $\boldsymbol{p}_{u'}$. Tällöin kukin näistä vektoreista voidaan laskea huomioimatta toisia, jolloin laskentaa pystytään hajauttamaan.

Painotukset matriisihajotelmissa

Matriisihajotelmiin perustuvissa toteutustavoissa lopputulosta voidaan edelleen parantaa lisäämällä laskentaan painotuksia. Koren, Bell ja Volinsky [23] huomauttavat, että käyttäjien arviot ohjelmista saattavat vaihdella paljonkin johtuen käyttäjistä tai ohjelmista itsestään. Tyypilliset yhteistoiminnallisen suosittelun lähdetiedot saattavat sisältää käyttäjiä, joilla on taipumus antaa keskivertokäyttäjää korkeampia tai matalampia arvosanoja ohjelmille. Samalla tavoin myös joillain ohjelmilla saattaa olla taipumus saada muita ohjelmia korkeampia tai matalampia arvosanoja.

Koska sisätulolla $\boldsymbol{q}_i{}^\intercal \boldsymbol{p}_u$ on vaikea ottaa huomioon esimerkiksi edellä mainitun kaltaisia ilmiöitä, suosittelujärjestelmä ilmaisee näitä ilmiöitä käyttäjä- tai ohjelmakohtaisilla painotuksilla (engl. bias). Käyttäjään u ja ohjelmaan i liittyvä painotus

$$\boldsymbol{B}_{ui} = \mu + \boldsymbol{b}_i + \boldsymbol{b}_u,$$

jossa μ on kaikkien arvioiden keskiarvo, $\boldsymbol{b}_i$ on ohjelman painotus ja $\boldsymbol{b}_u$ on käyttäjän painotus. Ohjelman painotus $\boldsymbol{b}_i$ kuvaa sitä, minkälaisia arvioita kyseinen ohjelma saa verrattuna muihin ohjelmiin, eli kehutaanko tai moititaanko sitä normaalia enemmän. Painotus $\boldsymbol{b}_u$ taas kuvaa sitä, kuinka kriittinen käyttäjä u on arvioijana. Kun painotukset otetaan kaavaan mukaan, tällöin ennusteeksi saadaan

$$\boldsymbol{P}_{ui} = \mu + \boldsymbol{b}_i + \boldsymbol{b}_u + \boldsymbol{q}_i{}^\intercal \boldsymbol{p}_u.$$

Tässä luvussa on esitelty mallipohjaiset yhteistoiminnalliset suosittelualgoritmit perusmuodossaan. Usein yhteistoiminnallisen suosittelun piirissä käytetään edistyneempiä matriisihajotelmatekniikoita. Niissä matriisin alkoiden arvojen ennustamisessa saatetaan hyödyntää esimerkiksi erilaisia todennäköisyysmalleja.

3.1.3 Kylmäkäynnistysongelma

Mikäli yhteistoiminnallisessa suosittelussa ei ole tietoa joko käyttäjän mieltymyksistä tai ohjelmista, saatetaan törmätä ongelmaan, jota kutsutaan nimellä kylmäkäynnistysongelma (engl. cold start problem) [23]. Kylmäkäynnistysongelma tarkoittaa siis tilannetta, jossa käyttäjästä on tarjolla hyvin vähän tai ei lainkaan informaatiota, eli hän on katsonut tai arvioinut hyvin vähän tai ei lainkaan ohjelmia. Tällöin

on hyvin vaikea tehdä päätelmiä käyttäjän mieltymyksistä. Sama koskee myös uusia ohjelmia. Jos ohjelma ei ole kerännyt yhtään arviota, kyseisen ohjelman suhdetta muihin ohjelmiin on vaikea määritellä. Lika, Kolomvatsos ja Hadjiefthymiades [26] jakavat kylmäkäynnistys-ongelman kolmeen luokkaan: a) suosittelut uusille käyttäjille, b) uusien ohjelmien suosittelu ja c) uusien ohjelmien suosittelu uusille käyttäjille.

Lika, Kolomvatsos ja Hadjiefthymiades esittävät omaksi ratkaisukseen kylmäkäynnistysongelmaan väestötieteellistä lähestymistapaa, jossa saman väestötieteellisen ryhmän jäsenten ajatellaan jakavan samankaltaiset mieltymykset. Lam et al [24] ehdottavat artikkelissaan samankaltaista lähestymistapaa ratkaisuksi kylmäkäynnistysongelmaan, missä käytetään hyväksi tietoja käyttäjän iästä, sukupuolesta ja ammatista. Yhteistä näissä ratkaisuissa on se, että käyttäjistä on tarjolla jonkinlaista taustatietoa. Tämä ei kuitenkaan päde Yle Areenan tapauksessa, joten menetelmää ei voida soveltaa.

3.1.4 Implisiittinen sisällön arviointi

Yhteistoiminnallinen suosittelu pohjautuu käyttäjien antamiin arvioihin ohjelmista tai sisältöobjekteista [12]. Luotettavin tapa kerätä laadukasta tietoa käyttäjien mieltymyksistä on kysyä tätä käyttäjiltä suoraan [19]. Esimerkiksi suoratoistopalvelu Netflixissä voi antaa arvioita katsomistaan ohjelmista, joskin Netflix käyttää suosittelussaan hyväkseen myös implisiittistä palautetta [2].

Usein eksplisiittistä tietoa käyttäjien ohjelma-arvioista ei kuitenkaan ole tarjolla. Esimerkiksi Yle Areenassa tai Lasten Areenassa käyttäjä ei voi nimenomaisesti ilmaista palvelussa, kuinka miellyttävä toistettu ohjelma oli. Tämän vuoksi on syytä turvautua implisiittiseen palautteeseen. Implisiittisessä palautteessa käyttäjien mieltymykset päätellään käyttäjän käyttäytymisestä palvelussa [19]. Implisiittisen palautteen keräilymuotoja saattavat olla käyttäjän ostoshistoria, selailuhistoria, haut tai jopa hiiren liikkeet. Esimerkiksi, käyttäjä, joka toisti paljon tietyn genren ohjelmia todennäköisesti pitää kyseisestä genrestä.

Hu, Koren ja Volinsky [19] luettelevat neljä implisiittiseen palautteeseen liittyvää erityispiirrettä. Ensimmäinen heidän havaintonsa on, ettei implisiittisesti voi antaa negatiivista palautetta. Edellä mainituista implisiittisistä palautetiedoista voi päätellä, mistä käyttäjä on kiinnostunut, mutta sen päättely on hankalaa, mistä käyttäjä ei ole kiinnostunut tai ei pidä. Jos esimerkiksi käyttäjä ei ole toistanut ohjelmaa i, johtuuko se siitä, ettei käyttäjä pitänyt i:sta vai siitä, ettei hän ole löytänyt palvelusta i:ta tai hän ei ole vielä ehtinyt katsella tai kuunnella sitä?

Toisen heidän havaintonsa mukaan implisiittinen palaute on luonnostaan hälyisää. Vaikka käyttäjien käyttäytymisestä kerätäänkin paljon tietoja, käyttäjien todellisia mielihaluja voidaan vain arvailla. Esimerkiksi, kun käyttäjä katsoi ohjelman i', pitikö hän siitä vai oliko hän kuitenkin lopulta pettynyt?

Kolmantena, eksplisiittisessä palautteessa palautteen arvoa voidaan pitää käyttäjän preferenssinä kyseistä sisältöobjektia kohtaan. Implisiittinen palaute sen sijaan kuvastaa luottamuksen määrää. Implisiittinen palaute saattaa olla esimerkiksi, kuinka suuren osuuden käyttäjä on toistanut ohjelmasta, kuinka usein käyttäjä on ostanut tietynlaista tuotetta ja niin edespäin. Mikäli tämän kaltaista implisiittistä palautetta on kertynyt paljon, voidaan muodostaa suurempi luottamus siihen, että käyttäjä on kiinnostunut kyseessä olevan kaltaisista sisältöobjekteista.

Viimeisenä havaintona Hu, Koren ja Volinsky korostavat implisiittiseen palautteeseen sopivien arviointimenetelmien käyttämistä. Eksplisiittisessä palautteessa virheen määrä ennusteen ja todellisen palautteen välillä voidaan laskea yksiselitteisesti. Implisiittisessä palautteessa toiminta ei ole näin suoraviivaista, vaan huomioon otettavia seikkoja ovat muun muassa sisältöobjektin saatavuus tai mahdolliset kilpailuasetelmat muiden sisältöobjektien kanssa.

Kun käyttäjän mieltymyksiä yritetään tulkita implisiittisestä palautteesta, edellä mainitun perusteella siihen liittyy huomattavan paljon epävarmuutta. Implisiittisten toimintojen (ohjelman toisto, hiiren klikkaus) tulkitseminen positiiviseksi palautteeksi on jo epävarmaa sinänsä, mutta negatiivisia palautteita syötteistä ei voida tulkita juuri minkäänlaisella varmuudella.

Hu, Koren ja Volinsky [19] esittelevät ratkaisunaan implisiittiseen arviointiin muutamia käsitteitä. Ensimmäinen käsite on $\boldsymbol{Pr}_{ui}$, joka kuvaa käyttäjän kiinnostusta tai mieltymystä sisältöobjektia kohtaan, englannin sanan *preference* mukaan.

$$\boldsymbol{Pr}_{ui} = \begin{cases} 1 & : \boldsymbol{Y}_{ui} > 0 \\ 0 & : \boldsymbol{Y}_{ui} = 0 \end{cases}$$

Eli jos käyttäjä u on toistanut ohjelmaa i, tällöin oletetaan, että käyttäjä on kiinnostunut ohjelmasta i. Jos taas käyttäjä ei ole katsonut ohjelmaa, tällöin oletetaan, ettei käyttäjällä ole mielipidettä ohjelmasta suuntaan tai toiseen.

Hu, Koren ja Volinsky esittelevät myös toisen suureen, joka kuvaa luottamusta arvioon $\boldsymbol{Pr}_{ui}$. Luottamus $\boldsymbol{C}_{ui}$ määritellään seuraavasti:

$$\boldsymbol{C}_{ui} = 1 + \alpha \boldsymbol{Y}_{ui},$$

missä α on luottamuksen kasvua määräävä vakio. Jos esimerkiksi käyttäjä u on katsonut ohjelmaa i paljon, tällöin luottamus arvoon $\boldsymbol{Pr}_{ui}$ kasvaa.

Kun edellä mainitut suureet otetaan mukaan, virhefunktioksi muodostuu

$$\mathcal{L}(\boldsymbol{q}, \boldsymbol{p}) = \sum_{ui} \boldsymbol{C}_{ui}(\boldsymbol{Pr}_{ui} - \boldsymbol{q}_i{}^{\intercal}\boldsymbol{p}_u)^2 + \lambda(\|\boldsymbol{q}_i\|^2 + \|\boldsymbol{p}_u\|^2).$$

3.2 Sisältöön perustuva suosittelu

Pazzani ja Billsus [29] kuvaavat artikkelissaan sisältöön perustuvat suosittelujärjestelmät järjestelmiksi, jotka suosittelevat sisältöobjektia käyttäjälle sisältöobjektin kuvauksen ja käyttäjän kiinnostuksen kohteiden perusteella. Sisältöön perustuvassa suosittelussa ei siis käytetä hyväksi muiden käyttäjien arvioita sisältöobjekteista. Sisältöön perustuva suosittelu analysoi sisältöobjektien kuvauksia tunnistaakseen sisältöobjekteja, jotka kiinnostavat käyttäjää.

Suositeltavat sisältöobjektit ovat tyypillisesti tallennettuina tietokantaan. Kukin sisältöobjekti muodostaa rivin tietokannassa. Tietokannan taulun sarakkeet ovat sisältöobjektien attribuutteja. Ravintoloiden tietokannan esimerkkitietoa nähdään taulukossa 2.

Taulukon 2 tietoja voisi käyttää esimerkiksi web-sivustolla, joka suosittelee käyttäjälle ravintoloita. Mikäli tässä kyseisessä esimerkissä käyttäjä pystyisi tekemään itselleen käyttäjäprofiilin, joka kuvaa, minkälaisista ravintoloista hän pitää, tällöin tietokannan ravintoloista voitaisiin tehdä poimintoja, mitkä ravintolat saattaisivat olla käyttäjälle mieluisia. Edellä kuvattujen strukturoitujen attribuuttien lisäksi sisältöobjekteja saatetaan kuvata myös vapaalla tekstillä. Esimerkiksi Yle Areenan audiosisällöissä on käytössä sisältöön perustuva suosittelu, missä ohjelmassa puhutut asiat on muutettu tekstiksi automaattisen puheentunnistuksen avulla. Teksti muutetaaan edelleen sanavektoreiksi. Nämä sanavektorit eli käytännössä ohjelmassa puhutut sisällöt toimivat kyseisen Yle Areenan audiosisällön kuvailuna, jonka perusteella suosittelua tehdään. Koska Yle Areenassa ei kerätä käyttäjistä eksplisiittisiä sisältöpreferenssejä, audiosuosittelua käytetään suosittelemaan jonkin audio-ohjelman yhteydessä samankaltaisia sisältöjä.

On olemassa tekniikka, jolla vapaa teksti voidaan esittää strukturoidussa muodossa. Tätä tekniikkaa käyttävät monet sisältöön perustuvat suosittelujärjestelmät [29]. Aluksi sanat muutetaan niin sanottuun perusmuotoon tai ”juurimuotoon” prosessilla, jota kutsutaan stemmaukseksi, karsinnaksi tai typistämiseksi (engl. stemming). Tässä prosessissa esimerkiksi sanan ”tietokone” erilaiset taivutusmuodot, kuten ”tie-

Taulukko 2: Ravintoloiden tietokanta.

ID	Nimi	Tyyppi	Palvelumuoto	Hintataso
10001	Pizzeria Milano	italialainen	Tiskiltä nouto	Edullinen
10002	Cafe de Lyon	ranskalainen	Pöytiin tarjoilu	Keskitaso
10003	Bistro de Michel	ranskalainen	Pöytiin tarjoilu	Kallis

tokoneen" tai "tietokoneeksi", typistetään yhdeksi juurimuodoksi. Kaikki tällaiseen juurimuotoon muutetut sanat muodostavat yhden termin (engl. term). Termien esiintyvyysuseutta dokumenttijoukossa mitataan tekniikalla, josta käytetään lyhennettä TF*IDF, joka muodostuu englannin sanoista term frequency, inverse document frequency. Suomeksi tätä voisi kutsua termitaajuus - käänteinen dokumenttitaajuus -suureeksi. TF, eli sanan tai termin taajuus dokumentissa on lukema, kuinka usein sana esiintyy yhdessä dokumentissa. Mikäli sana esiintyy usein tietyssä dokumentissa, voidaan päätellä, että kyseinen sana on merkittävä kyseiselle dokumentille. DF, eli dokumenttitaajuus on lukema, kuinka usein sana esiintyy kokoelmassa dokumentteja. Mikäli sana esiintyy usein laajassa kokoelmassa dokumentteja, voidaan päätellä, että sana on ylipäänsä yleinen. Tällaisia sanoja saattavat olla esimerkiksi suomen kielen sanat "ja", "on" ja niin edespäin. Kun lasketaan jakolasku $\frac{TF}{DF}$, eli $TF * IDF$, joka siis kuvaa sanan t tärkeyttä dokumentissa d, tällöin vain dokumentille oikeasti merkittävät sanat saavat korkean painoarvon ja kielessä yleisesti usein esiintyvät sanat sen sijaan eivät.

Tällä tavoin saadaan selville kunkin sanan merkittävyyden määrä kyseiselle sisällölle, eli missä määrin kukin sana kuvaa sisältöä. Pazzani ja Billsus huomauttavat, että sanan käyttökontekstia ei tässä oteta huomioon. Mikäli esimerkiksi ravintola-arvostelussa sanotaan: "tässä ravintolassa ei ollut mitään vegaaneille", jää huomiotta se tosiasia, että ravintola nimenomaan *ei* ole vegaaneille sopiva vaihtoehto kyseisen arvion mukaan.

Useat sisältöön perustuvat suosittelutavat taltioivat tietoja käyttäjistään käyttäjäprofiileihin [29]. Pazzani et al luettelevat kaksi asiaa huomionarvoisina piirteinä, joita käyttäjäprofiileihin taltioidaan:

1. Käyttäjäprofiiliin taltioituu käyttäjän mieltymyksien malli, eli kuvaus siitä, minkälaiset sisältöobjektit miellyttävät käyttäjää. Eräs yleinen tapa kuvata tätä on muodostaa funktio, joka ennustaa sitä, missä määrin kukin sisältöobjekti kiinnostaa käyttäjä.

2. Käyttäjäprofiiliin taltioituu myös käyttäjän ja järjestelmän välisen vuorovaikutuksen historia. Tähän käyttäjähistoriaan saatetaan taltioida tietoa esimerkiksi sii-

tä, mitä sisältöobjekteja ja mitä muuta informaatiota käyttäjä katseli samaan aikaan. Käyttäjähistoriaan saatetaan myös taltioida esimerkiksi käyttäjän käyttämiä hakusanoja.

Samoin kuin yhteistoiminnallisessa suosittelussa, myös sisältöpohjaisessa suosittelussa voidaan luoda käyttäjistä malli, jonka perusteella sisältöjä suositellaan käyttäjille. Mallin luominen voidaan nähdä luokitteluoppimisena (engl. Classification Learning). Malli opetetaan luokittelemaan sisältöobjektit kahteen luokkaan: ”miellyttää käyttäjää” ja ”ei miellytä käyttäjää”. Mallin opettamisen lähdetietona ovat joko käyttäjän eksplisiittiset arviot sisältöobjekteista tai käyttäjän toiminta kyseisessä palvelussa, jonka pohjalta kerätään implisiittisiä arvioita sisältöobjekteista. Jos esimerkiksi käyttäjä osti jonkin tuotteen, tällöin voidaan tehdä päätelmä, että käyttäjä piti tuotteesta. Jos taas käyttäjä osti tuotteen ja palautti sen, tällöin voidaan tehdä päätelmä, että käyttäjä ei pitänyt tuotteesta.

Pazzani ja Billsus luettelevat artikkelissaan algoritmeja, joilla voidaan toteuttaa edellä mainittu luokittelumalli. Tässä mainitaan niistä muutamia.

Päätöspuut ja sääntöpohjainen päättely

Päätöspuualgoritmeissa (engl. Decision Tree) luodaan puu, jossa lähdetieto jaetaan puumaisesti kategorioihin ja alakategorioihin [29, 27]. Puun jakautumista jatketaan, kunnes yhdessä alakategoriassa on vain samankaltaisia sisältöobjekteja. Lähdetieto jaetaan puussa osiin vertailemalla valittuja attribuutteja. Esimerkiksi tekstin luokittelussa luokitteluperusteena voisi olla jonkin sanan olemassaolo tekstissä.

Lähimmän naapurin luokittelumenetelmät

Lähimmän naapurin luokittelumenetelmät taltioivat kaiken mallin oppimiseen käytettävän lähdetiedon muistiin. Kun uusi sisältöobjekti luokitellaan, tällöin algoritmi vertaa sisältöobjektia muihin, mallin jo tuntemiin sisältöobjekteihin ja määrittelee sen ”lähimmän naapurin” (engl. nearest neighbour) tai k kappaletta lähimpiä naapureita [29, 27]. Tällöin uuden sisältöobjektin luokitus on sen lähimpien naapureiden luokitus.

Luokittelu hypertasojen avulla

Lineaarisissa luokittelijoissa muodostetaan moniulotteisessa avaruudessa niin sanottuja hypertasoja, joilla erotetaan kyseisen hyperavaruuden osia toisistaan, jolloin si-

sältöjä saadaan jaettua luokkiin [29, 27]. Esimerkiksi tukivektorikone (engl. Support Vector Machine) on toimii tällä periaatteella.

3.3 Muut suosittelualgoritmit

Tässä luvussa käydään lyhyesti läpi yhteistoiminnallisen suosittelun ja sisältöön perustuvan suosittelun lisäksi muita sisältöjen suosittelutapoja.

Yhteistoiminnallisen suosittelun ja sisältöön perustuvan suosittelun lisäksi Burke [12] luettelee suosittelumenetelminä myös väestötieteellisen suosittelun, hyötyyn perustuvan suosittelun sekä käyttäjätietoon perustuvan suosittelun.

Väestötieteellinen suosittelu

Väestötieteellisessä suosittelussa käyttäjät pyritään luokittelemaan heidän attribuuttiensa perusteella ja tekemään sisältösuosituksia heidän väestötieteellisten luokkiensa perusteella. Burke mainitsee esimerkkinä Grundy-järjestelmän vuodelta 1979, joka suositteli käyttäjille kirjoja. Suosittelu tehtiin käyttäjäkyselyssä kerättyjen vastausten perusteella, joita vertailtiin käsin koostettuihin niin sanottuihin stereotyyppikäyttäjiin. Väestötieteellinen suosittelu muodostaa ihmiseltä ihmiselle -korrelaatioita yhteistoiminnallisen suosittelun tapaan. Väestötieteellisen lähestymistavan etuna on se, että käyttäjän arvioiden kerääminen sisältöobjekteista ei ole välttämätöntä. Väestötieteellinen suosittelu ei kuitenkaan ole toteutustapansa vuoksi kovin personoitua.

Hyötyyn perustuva suosittelu

Hyötyyn perustuvassa suosittelussa vertaillaan käyttäjän tarpeita ja tarjolla olevia vaihtoehtoja. Eri järjestelmissä keskeisen kysymyksen ratkaisu, eli hyödyllisyyden määrittelevän funktion luominen on toteutettu eri tavoilla.

Esimerkiksi, oletetaan, että käyttäjä on vaatteita myyvässä verkkokaupassa, jossa käyttäjälle suositellaan vaatteita hyötyyn perustuvan suosittelun avulla. Tällöin suositeltaville tuotteille lasketaan hyödyllisyys perustuen käyttäjän mieltymyksiin. Esimerkiksi jokin myytävä vaatekappale saattaa saada valmistajansa vuoksi kaksinkertaisen hyödyn, hintansa vuoksi 1,5-kertaisen hyödyn ja valmistusmateriaalin vuoksi 1-kertaisen hyödyn. Tämän jälkeen tuotteen hyödyllisyys kyseiselle käyttäjälle voidaan laskea, jolloin suositeltavat tuotteet voidaan järjestää kyseiselle käyttäjälle hyödyllisyysjärjestykseen.

Käyttäjätietoon perustuva suosittelu

Käyttäjätietoon perustuvassa suosittelussa suositellaan sisältöobjekteja käyttäjän tarpeisiin perustuen. Käyttäjätietoon perustuvien järjestelmien toiminta perustuu tarkkaan tietoon tuotteista, käyttäjien mieltymyksistä ja suosittelun kriteereistä. Käyttäjätietoon perustuvan suosittelun merkittävin ero muihin suosittelutapoihin nähden on niiden tieto siitä, miten tietty sisältöobjekti kohtaa tietyn käyttäjän tarpeet. Tämän tiedon perusteella järjestelmä voi päätellä suhteen käyttäjän ja sisältöobjektin välille ja tehdä suosituksen sen perusteella.

Esimerkiksi asuntoja välittävä verkkosivusto saattaa järjestää myytävät asunnot käyttäjätietoon perustuvan suosittelun avulla. Tässä esimerkissä käyttäjältä kysytään suoraan asunnon minimi- ja maksimihintaa, haluttua pinta-alaa, huoneiden lukumäärää ja niin edelleen. Näiden, käyttäjän eksplisiittisesti antamien mieltymysten avulla myytävät asunnot voidaan järjestää käyttäjälle relevanttiin järjestykseen.

4 Tutkimuskohteet

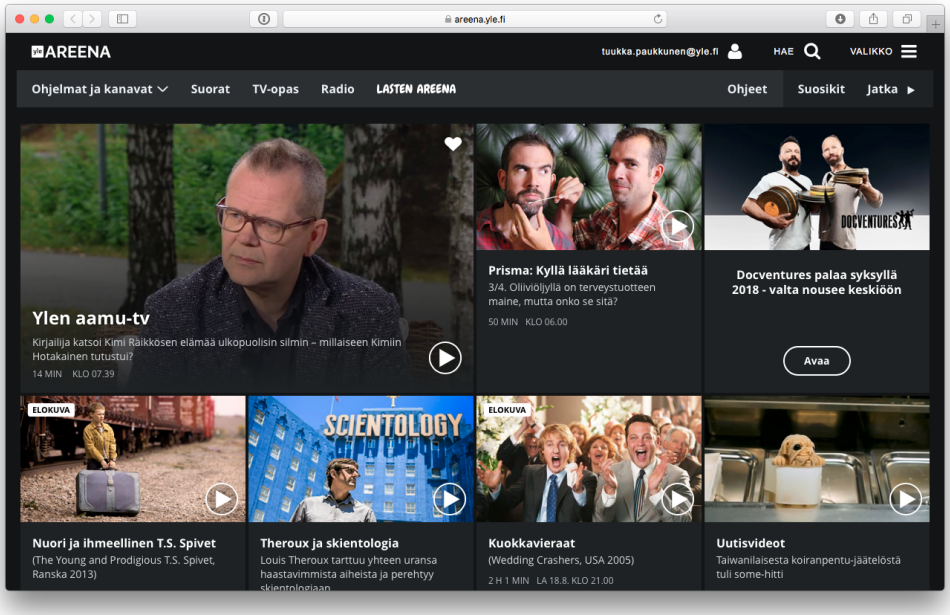
Tässä luvussa esitellään tämän tutkielman tutkimuskohteet, joilla lasten suositteluja tuotetaan. Ensimmäisenä tutkimuskohteena on Yle Areenan suosittelujärjestelmä. Toisena tutkimuskohteena on tätä tutkielmaa varten tehty muistipohjainen suosittelujärjestelmä, josta käytetään myös nimitystä muistipohjainen suosittelija.

4.1 Yle Areenan suosittelujärjestelmä

Tässä luvussa esitellään Yle Areenan suosittelujärjestelmä ja sen toimintaperiaatteet. Yle Areenan suosittelujärjestelmästä käytetään myös nimitystä Areena-recommender. Yle Areenan suosittelujärjestelmän ovat toteuttaneet Ylen suosittelutiimin kehittäjät.

Pääperiaatteet

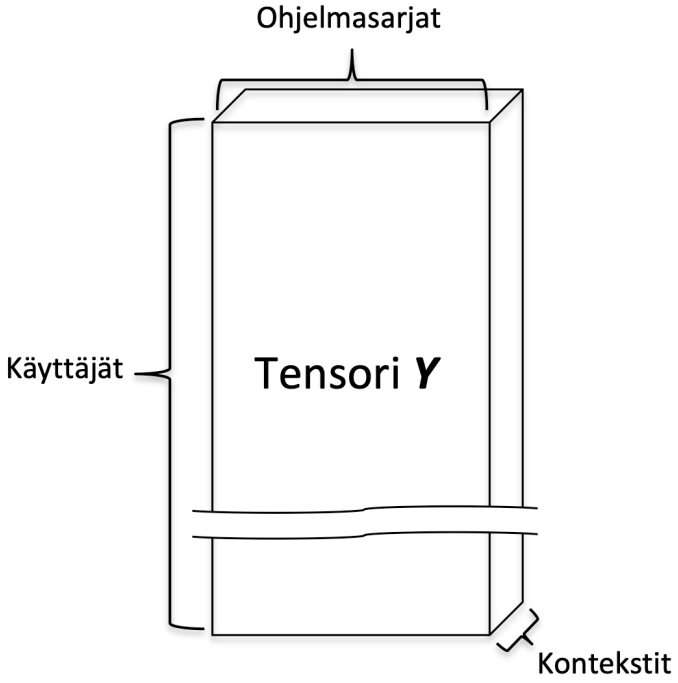
Areenan etusivulla näytetään käyttäjälle suosituksia 0-11 ensimmäisessä ”laatikossa”. Areenan toimitus täyttää käsin tehtävillä ohjelmasuosituksilla osan näistä laatikoista. Tämän jälkeen Areenan suosittelujärjestelmä täyttää loput ohjelmasuositukset. Kuvassa 2 nähdään esimerkki, jossa ensimmäisenä suositeltavana ohjelmana on Ylen aamu-tv, jota seuraavat muun muassa Prisma ja Docventures.



Kuva 2: Areenan etusivun ohjelmasuosituksia.

Samalla toimintaperiaatteella toimivat myös kategoriakohtaiset suosittelut (esimerkiksi https://areena.yle.fi/tv/ohjelmat/sarjat). Lastenohjelmien suosittelu on tutkielman kirjoitushetkellä rajattu pois Areenan suosittelujärjestelmästä.

Suosittelun lähdetiedoiksi on kerätty Areenan ohjelmien katselu- ja kuuntelutiedot eli toistotiedot 50 vuorokauden ajalta. Toistotiedot koostuvat csv-muotoon taltioiduista tietokantariveistä, joissa sarakkeina on joko selaimen tai mobiililaitteen tunniste, ohjelman tunniste, ohjelmasta toistettu millisekuntimäärä, ohjelman kesto millisekunteina sekä aikaleima, jolloin toistaminen oli aloitettu. Näiden lisäksi lähdetietoina ovat myös edellä mainitun ajanjakson aikana toistettujen ohjelmien metatiedot. Näistä tiedoista on koostettu tensori $\boldsymbol{Y}$, jonka arvo $\boldsymbol{Y}_{uci}$ sisältää tiedot siitä, onko kukin muutamasta miljoonasta käyttäjästä u katsonut jonkin kymmenistä tuhansista ohjelmista i joko mobiili- tai selainkäyttökontekstissa c vai ei. Eli kutakin käyttäjää u kohden tensorissa on yksi ulottuvuus, kutakin ohjelmaa i kohden on toinen ulottuvuus ja kutakin käyttökontekstia c kohden on kolmas ulottuvuus. Mikäli käyttäjä u on katsonut ohjelman i käyttökontekstissa c, tällöin tensorin $\boldsymbol{Y}$ ensimmäisen ulottuvuuden u, toisen ulottuvuuden i ja kolmannen ulottuvuuden c solussa on 1, muutoin solussa on 0. Piirros tensorista $\boldsymbol{Y}$ nähdään kuvassa 3.



Kuva 3: Lähdetietojen tensori $\boldsymbol{Y}$. Tensorin mittasuhteet ovat viitteellisiä.

Keskeinen käsite lähdetiedoissa on ohjelman toisto (engl. view). Mikäli käyttäjä on toistanut ohjelmasta vähintään 40 %, tällöin tämä katsotaan ohjelman toistoksi. Toiston arvo on binäärinen eli joko 0, mikäli ohjelmaa ei ole toistettu tai 1, mikäli

ohjelmasta on toistettu vähintään 40 %. Saman käyttäjän ja saman ohjelman toistokerroista huomioidaan vain ensimmäinen, eli mahdolliset useat saman ohjelman toistokerrat jätetään huomiotta. 50 vuorokauden ajalta ohjelmien toistoja kerääntyy noin 30 miljoonaa kappaletta. Kun tarkastellaan kokonaisten sarjojen toistoja, näitä kerääntyy 50 vuorokauden ajalta noin 13,5 miljoonaa kappaletta. Koska tensorissa $\boldsymbol{Y}$ solujen lukumäärä on yli 7 biljoonaa, tensorissa on ykkösiä noin 0,00018 % eli tensori on varsin harva. Areenassa ei ole mahdollista arvioida ohjelmia eksplisiittisesti, joten tässä tapauksessa ohjelman toisto toimii implisiittisenä tietona, että käyttäjä on ainakin jossain määrin kiinnostunut kyseessä olevasta ohjelmasta.

Hierarkinen Poisson-matriisihajotelma

Gopalan, Hofman ja Blei esittelivät hierarkisen Poisson-matriisihajotelmamenetelmän (engl. Hierarchical Poisson matrix factorization, HPF) vuonna 2015 [17].

HPF-menetelmässä luodaan todennäköisyysmalli, joka pyrkii ennustamaan, millä todennäköisyydellä käyttäjä u kuluttaisi sisältöä i. Mallissa luodaan matriisihajotelmien tapaan vektori $\boldsymbol{p}_u$ kutakin käyttäjää kohden ja vektori $\boldsymbol{q}_i$ kutakin sisältöobjektia kohden. Vektorit sisältävät K-kappaletta piileviä tekijöitä eli vektorit ovat K-ulotteisia. HPF-mallissa oletetaan, että matriisissa tai tensorissa $\boldsymbol{Y}$ kukin solu on poiminta Poisson-jakaumasta. Kunkin solun arvo on vektorien $\boldsymbol{p}_u$ ja $\boldsymbol{q}_i$ sisätulo, eli

$$\boldsymbol{Y}_{ui} \sim \text{Poisson}(\boldsymbol{p}_u^\intercal \boldsymbol{q}_i).$$

HPF-mallissa käyttäjän aktiivisuus on Gamma-jakautunut

$$\xi_u \sim \text{Gamma}(\kappa_u^{shp}, \kappa_u^{rte}).$$

Yläindeksit *shp* ja *rte* tarkoittavat Gamma-jakauman parametreja, joista käytetään englanniksi nimityksiä ”shape” ja ”rate”. Käyttäjän piilevien tekijöiden vektori noudattaa myös Gamma-jakaumaa

$$\boldsymbol{p}_u \sim \text{Gamma}(\gamma_u^{shp}, \gamma_u^{rte}).$$

Samalla tavoin sisältöobjektin suosion taso on Gamma-jakautunut

$$\eta_i \sim \text{Gamma}(\tau_i^{shp}, \tau_i^{rte}).$$

Sisältöobjektin piilevien tekijöiden vektori noudattaa myös Gamma-jakaumaa

$$\boldsymbol{q}_i \sim \text{Gamma}(\lambda_i^{shp}, \lambda_i^{rte}).$$

Ennuste käyttäjän kiinnostuksen tasosta sisältöobjektia kohtaan on Poisson-jakautunut

$$\boldsymbol{Y}_{ui} \sim \text{Poisson}(\boldsymbol{p}_u^\intercal \boldsymbol{q}_i).$$

Gopalanin, Hofmanin ja Blein mukaan tällainen hierarkinen rakenne kuvastaa hyvin sitä moninaisuutta, missä jotkut käyttäjät kuluttavat sisältöjä enemmän kuin toiset ja jotkut sisältöobjektit ovat suositumpia kuin toiset.

Yllä mainitut Gamma-jakaumien parametrit alustetaan arvoilla seuraavasti. Parametrit κ^{rte}, τ^{rte}, γ^{rte}, γ^{shp}, λ^{rte} ja λ^{shp} alustetaan pienillä satunnaisluvuilla. Näiden lisäksi:

$$\kappa_u^{shp} = a' + Ka,$$

$$\tau_i^{shp} = c' + Kc,$$

jossa parametrien a, a', c ja c' arvoiksi asetetaan $0,3$. K on piilevien tekijöiden lukumäärä, jonka sovelluksen toteuttajat valitsevat.

Edellä mainittujen lisäksi Gopalan, Hofman ja Blei esittelevät muuttujan

$$\boldsymbol{z}_{uik} \sim \text{Poisson}(\boldsymbol{p}_u^\intercal \boldsymbol{q}_i).$$

Muuttujan $\boldsymbol{z}_{uik}$ kunkin arvon k voidaan ajatella kuvastavan komponentin k kontribuutiota havaintoon $\boldsymbol{Y}_{ui}$. Kun muuttujien $\boldsymbol{z}_{uik}$ arvot summataan k:n yli, saadaan tulokseksi $\boldsymbol{Y}_{ui}$. Muuttujien $\boldsymbol{z}_{uik}$ Poisson-jakaumien summa on multinomiaalisesti jakautunut, eli

$$\boldsymbol{z}_{ui} \sim \text{Mult}(\boldsymbol{Y}_{ui}, \phi_{ui}).$$

Muuttuja $\boldsymbol{z}_{ui}$ on mukana helpottamassa algoritmin laskentaa ja kuvaamista.

Mallin opetus noudattaa seuraavaa algoritmia, jota toistetaan, kunnes konvergoituminen tapahtuu.

1. Jokaista käyttäjä/sisältöobjektiparia kohden, jossa $\boldsymbol{Y}_u i > 0$, päivitetään parametrin ϕ_{ui} arvo:

$$\phi_{ui} \propto \exp(\Psi(\gamma_{uk}^{shp}) - \log \gamma_{uk}^{rte} + \Psi(\lambda_{ik}^{shp}) - \log \lambda_{ik}^{rte}).$$

Yllä Ψ on digamma-funktio, joka on Gamma-funktion logaritmin ensimmäinen derivaatta. Tämä on peräisin Gamma-jakautuneen muuttujan logaritmin odotusarvosta, esimerkiksi

$$E[\log \boldsymbol{p}_{uk}] = \Psi(\gamma_{uk}^{shp}) - \log \gamma_{uk}^{rte}.$$

2. Jokaista käyttäjää kohden, päivitetään painoarvon ja aktiivisuuden parametrit:

$$\begin{aligned}
\gamma_{uk}^{shp} &= a + \sum_i \boldsymbol{Y}_{ui}\phi_{uik}, \\
\gamma_{uk}^{rte} &= \frac{\kappa_u^{shp}}{\kappa_u^{rte}} + \sum_i \frac{\lambda_{ik}^{shp}}{\lambda_{ik}^{rte}}, \\
\kappa_u^{rte} &= \frac{a'}{b'} + \sum_k \frac{\gamma_{uk}^{shp}}{\gamma_{uk}^{rte}}.
\end{aligned}$$

3. Jokaista sisältöobjektia kohden, päivitetään painoarvon ja suosion parametrit:

$$\begin{aligned}
\lambda_{ik}^{shp} &= c + \sum_u \boldsymbol{Y}_{ui}\phi_{uik}, \\
\lambda_{ik}^{rte} &= \frac{\tau_i^{shp}}{\tau_i^{rte}} + \sum_u \frac{\gamma_{uk}^{shp}}{\gamma_{uk}^{rte}}, \\
\tau_i^{rte} &= \frac{c'}{d'} + \sum_k \frac{\lambda_{ik}^{shp}}{\lambda_{ik}^{rte}}.
\end{aligned}$$

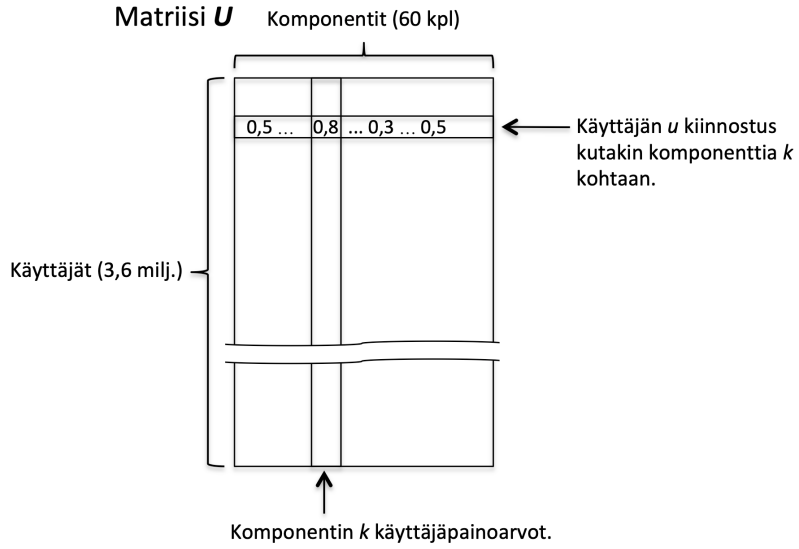
Yllä parametrien b' ja d' arvoiksi asetetaan 1.

Mallia opetetettaessa kutakin parametria optimoidaan vuorollaan pitäen muut parametrit vakioina. Toisin sanoen mallin opettaminen tapahtuu ALS-menetelmää käyttäen, joka esiteltiin luvussa 3.1.2. HPF-algoritmi pysäytetään, kun muutos todennäköisyyden logaritmissa (engl. log likelihood) on alle 0,00001 %.

HPF-menetelmän soveltaminen Areenan suosittelujärjestelmässä

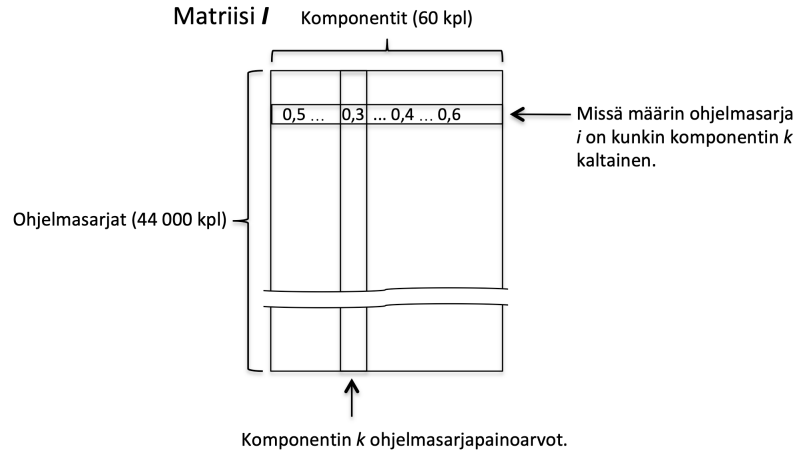
Areenan suosittelujärjestelmä noudattaa edellä kuvattua toimintatapaa pääpirteissään. Areenassa ennuste $\boldsymbol{P}$ tensorista $\boldsymbol{Y}$ mallinnetaan kolmesta matriisista $\boldsymbol{U}$, $\boldsymbol{I}$ ja $\boldsymbol{C}$.

Käyttäjille on matriisi $\boldsymbol{U}$, jossa $\boldsymbol{U}_{uk}$ kuvaa käyttäjärivin u saraketta k. Sarakkeet kuvaavat luvussa 3.1.2 mainittuja piileviä tekijöitä tai Areenan termein komponentteja. Areenan suosittelumallissa näitä on 60. Kukin komponentti kuvaa painoarvoa, missä määrin käyttäjä u on kiinnostunut komponentin k mukaisesta sisällöstä. Esimerkiksi jos käyttäjä u pitää urheilusta, tällöin rivin $\boldsymbol{U}_u$ urheilua vastaava sarake k sisältää korkean arvon. Matriisi $\boldsymbol{U}$ nähdään kuvassa 4.



Kuva 4: Käyttäjien matriisi $\boldsymbol{U}$. Mittasuhteet ovat viitteellisiä.

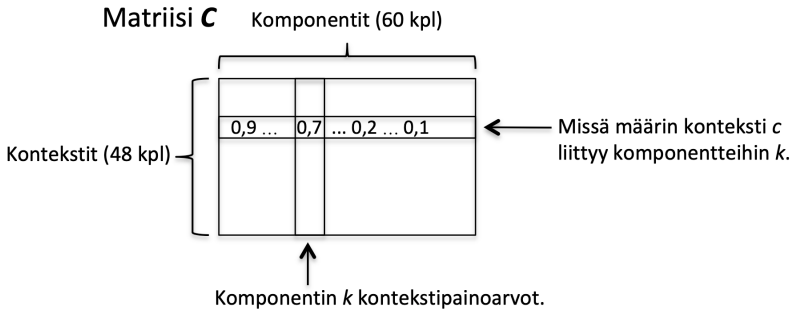
Ohjelmille on matriisi $\boldsymbol{I}$, jossa on rivi kullekin ohjelmalle i sekä sarake 60 komponentille k. Komponentit ohjelmien yhteydessä tarkoittavat sitä, missä määrin ohjelma i käsittelee komponentin k mukaista sisältöä. Matriisi $\boldsymbol{I}$ nähdään kuvassa 5.



Kuva 5: Ohjelmasarjojen matriisi $\boldsymbol{I}$. Mittasuhteet ovat viitteellisiä.

Kolmas matriisi käsittelee käyttökonteksteja. Käyttökonteksti tässä yhteydessä tarkoittaa vuorokauden tuntia ja joko selainta tai mobiililaitetta. Kontekstimatriisi $\boldsymbol{C}$

koostuu siis 48 rivistä eli yksi rivi kullekin vuorokauden tunnille kahta erilaista päätelaitetta (selain tai mobiililaite) kohden. Myös kontekstimatriisissa $\boldsymbol{C}$ on sarake kutakin komponenttia k kohden. Mikäli kontekstimatriisissa $\boldsymbol{C}$ rivin c sarakkeen k arvo on korkea, tällöin tämä tarkoittaa sitä, että komponentti k liittyy vahvasti käyttökontekstiin c. Matriisi $\boldsymbol{C}$ nähdään kuvassa 6.



Kuva 6: Kontekstien matriisi $\boldsymbol{C}$. Mittasuhteet ovat viitteellisiä.

Silmämääräisesti arvioiden komponentit Areenan suosittelussa muodostavat loogisia kokonaisuuksia. Areenan suosittelujärjestelmän hallintanäkymässä kuvassa 7 komponentti 0 sisältää lastenohjelmia, kun taas komponentissa 1 korostuu rikosdraamojen osuus. Kuvassa näytetään kustakin komponentista ohjelmasarjat, joissa kyseisen komponentin painoarvo on suurin.

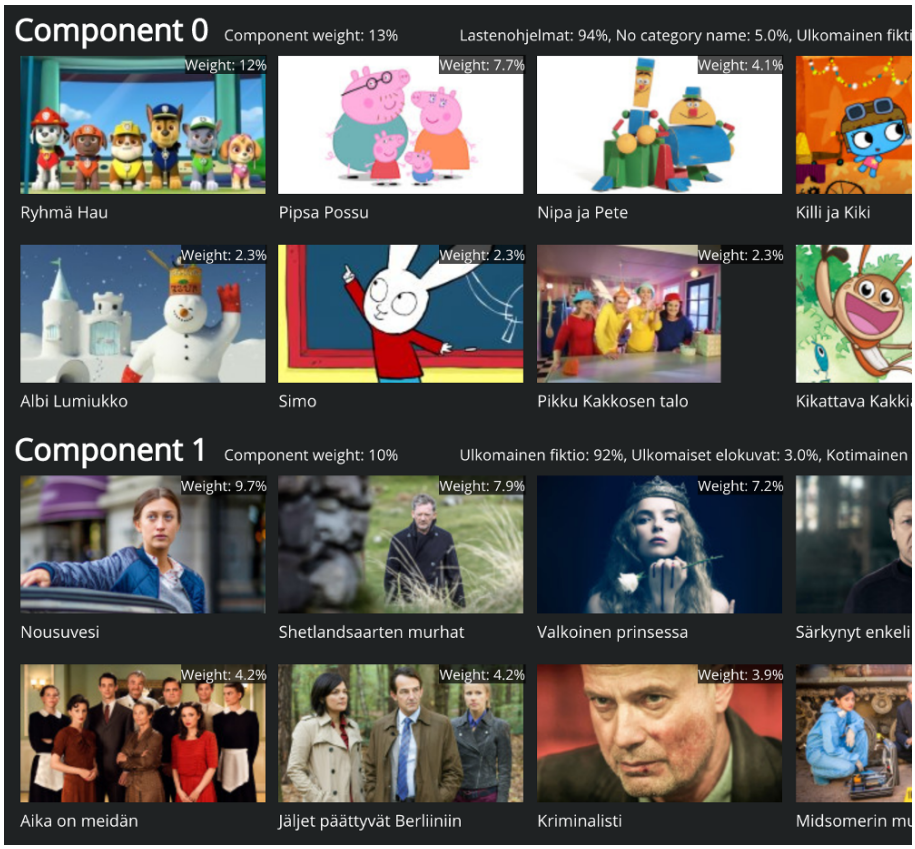
Areenan suosittelussa käyttäjien u, ohjelmien i ja kontekstien c kunkin komponentin arvo k oletetaan gamma-jakaumaksi. Se kuvaa odotusarvoa sille, kuinka oleellinen komponentin k aihe on kyseiselle käyttäjälle, kontekstille tai ohjelmalle, Gopalanin, Hofmanin ja Blein artikkelin mukaisesti [17]. Malli on

$$\begin{aligned}\boldsymbol{U}_{uk} &\sim \text{Gamma}(\alpha_{uk}, \beta_{uk}),\\ \boldsymbol{C}_{ck} &\sim \text{Gamma}(\alpha_{ck}, \beta_{ck}),\\ \boldsymbol{I}_{ik} &\sim \text{Gamma}(\alpha_{ik}, \beta_{ik}).\end{aligned}$$

Näiden gamma-jakaumien voidaan ajatella kuvaavan komponentin k mahdollista painoarvojakaumaa käyttäjä-, ohjelma- ja käyttökontekstikohtaisesti. Gamma-jakauman käsite kuvaillaan liitteessä 1.

Käyttäjän u ennustetaan katsovan ohjelma i kontekstissa c tasolla

$$\boldsymbol{\Lambda}_{uci} = \sum_k \boldsymbol{U}_{uk}\boldsymbol{C}_{ck}\boldsymbol{I}_{ik}. \in \mathbb{R}_+.$$



Kuva 7: Areenan komponenttien sisältöä.

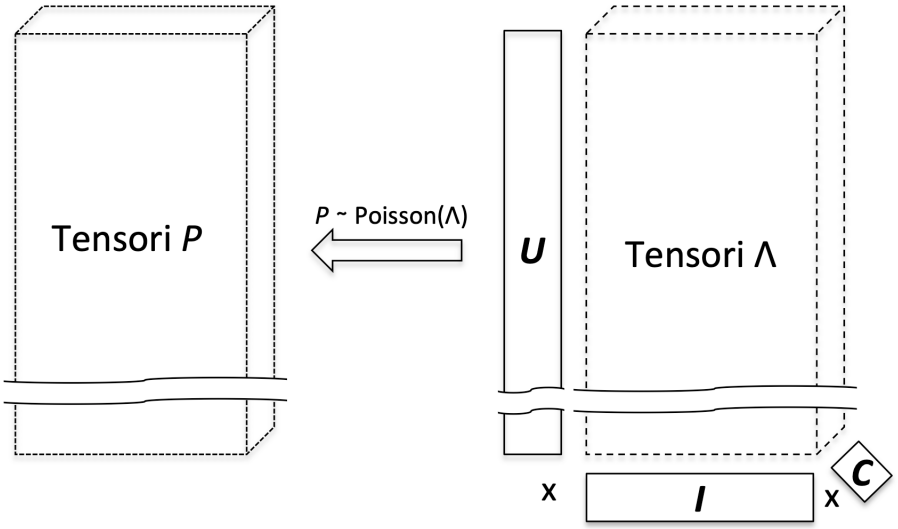
Tensorin $\boldsymbol{\Lambda}$ käyttäjä-, ohjelma- ja kontekstikohtaisen luvun voidaan tulkita kuvaavan sitä, minkä verran käyttäjä u on katsonut ohjelmaa i kontekstissa c. Luku saattaa olla esimerkiksi ”kerran 50 vuorokaudessa” tai ”0,1 kertaa 50 vuorokaudessa”. Luku ei suoraan kuvaa toistojen määrää ajanjaksolla, mutta sen voidaan kuitenkin ajatella olevan suure, joka kuvaa kyseessä olevan ohjelman toiston tasoa, jonka avulla ohjelmia voidaan asettaa järjestykseen. Ohjelma voi siis olla esimerkiksi ”suuresti toistettu” tai ”vain vähän toistettu”.

Ennuste käyttäjän u kiinnostuksesta ohjelmaa i kohtaan käyttökontekstissa c oletetaan Poisson-jakautuneeksi. Poisson-jakauma esitellään liitteessä 2.

$$\boldsymbol{P}_{uci} \sim \text{Poisson}(\boldsymbol{\Lambda}_{uci}).$$

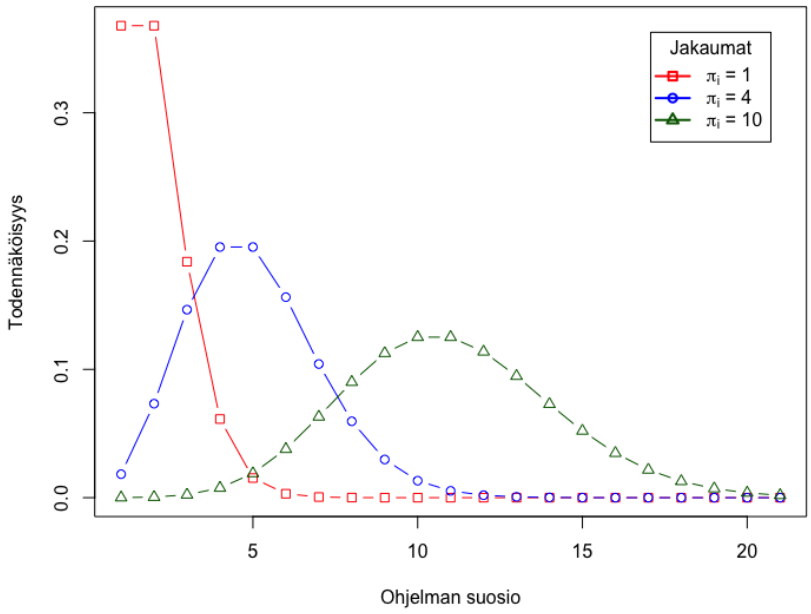
Kuva 8 havainnollistaa, miten matriiseista $\boldsymbol{U}$, $\boldsymbol{I}$ ja $\boldsymbol{C}$ muodostetaan tensori $\boldsymbol{\Lambda}$, jonka jälkeen tensorin $\boldsymbol{P}$ arvot muodostetaan poimimalla Poisson-jakaumista satunnaismuuttujan arvoja käyttäen kutakin parametria $\boldsymbol{\Lambda}_{uci}$.

Tosiasiassa sovellus ei taltioi tensoreita $\boldsymbol{\Lambda}$ tai $\boldsymbol{P}$ kokonaisina, vaan arvoja lasketaan ajonaikaisesti tarpeen mukaan.



Kuva 8: Tensorien $\boldsymbol{\Lambda}$ ja $\boldsymbol{P}$ muodostaminen. Mittasuhteet ovat viitteellisiä.

Merkitään $\pi_i = \sum_{uc} \boldsymbol{\Lambda}_{uci}$ ohjelman i toiston odotettavissa olevaa määrää kuvaavaksi suureeksi. Kuvassa 9 nähdään ohjelmien suosioiden Poisson-jakaumia, joissa parametrina on erilaisia ohjelmien toistomääriä. Jos ohjelman toistomäärä on pieni, eli $\pi_i = 1$, tällöin ohjelman suosio todennäköisesti ei ole suuri. Jos ohjelman taas ohjelman toistomäärä on suurempi, eli $\pi_i = 10$, tällöin ohjelma saattaa olla suosittu.



Kuva 9: Ohjelmien toistomäärien Poisson-jakaumat eri suosioiden perusteella.

Ennen mallin sovittamisen aloittamista matriisien $\boldsymbol{U}$, $\boldsymbol{C}$ ja $\boldsymbol{I}$ gamma-jakaumien pa-

rametrit alustetaan vakioarvoilla. Matriisien $\boldsymbol{U}$ ja $\boldsymbol{C}$ arvot kullakin sarakkeella k muokataan siten, että kunkin sarakkeen k summa on 1. Mallia sovitettaessa parametrit sovitetaan käyttäjä-, ohjelma- ja käyttökontekstikohtaisesti siten, että ennustetensorin $\boldsymbol{P}$ ja todellisen toistotietotensorin $\boldsymbol{Y}$ tunnettujen arvojen välinen virhe olisi mahdollisimman pieni. Mallin sovittaminen kuvailtiin tarkemmin edellä tämän luvun kappaleessa ”Hierarkinen Poisson-matriisihajotelma”.

4.1.1 Areenan suosittelun lisäykset Poisson-matriisihajotelmaan

Kun otetaan huomioon Areenan lähdetietojen tarvitsemat toistotietojen muokkausvaatimukset ja Areenan käyttötarkoitus, Poisson-matriisihajotelmassa on puutteita, joiden vuoksi suosittelujärjestelmään on tehty seuraavia lisäyksiä.

Sarjojen aggregointi

Yle Areenan suosittelujärjestelmässä on tehty päätös, että järjestelmä suosittelee sisältöjä sarjatasolla. Sarjojen suosittelu on Ylen liiketoiminnassa todettu luontevammaksi tavaksi suositella Yle Areenan sisältöjä yksittäisten jaksojen suosittelun sijaan.

Mikäli Areenan suosittelussa suositeltavaksi nousee jokin sarjan jakso, eli ohjelma i, tällöin suositellaan koko sarjaa s yksittäisen jakson sijaan. Areenan suosittelussa lasketaan sarjan jaksojen suosioiden tasot yhteen siten, että ne ovat edelleen vertailukelpoisia pisteohjelmien kanssa. Termillä ”pisteohjelma” tarkoitetaan sarjaa, jossa on vain yksi jakso.

Esimerkiksi, oletetaan, että ohjelmajoukko $\{i_1, i_2, ..., i_n\}$ vastaa yhtä sarjaa. Tällöin matriisissa $\boldsymbol{I}$ yksittäisiä jaksoja kuvaavat rivit $\{i_1, i_2, ..., i_n\}$ poistetaan ja lisätään tilalle yksi rivi sarjalle s.

Suosittelun suorituskyvyn parantaminen uusien ohjelmien osalta

Areenan suosittelumallin sovittamiseen ja tietojen siirtoon kuluu aikaa noin kolme tuntia. Tästä seuraa se, että suosittelu ei ota ollenkaan huomioon alle 3 tuntia vanhoja ohjelmia ja sen lisäksi suosittelu on painottunut koko 50 vuorokauden aikaikkunan ohjelmien suositteluun, jolloin tuoreita ohjelmia ei nouse suositeltujen ohjelmien joukkoon Yleisradion liiketoiminnan haluamalla tavalla.

Jotta myös alle 3 tuntia sitten julkaistut ohjelmat tulevat otetuksi huomioon suosittelussa, Areenan suositteluun on tehty keino, jolla viimeisen 24 tunnin aikana

julkaistuja ohjelmia painotetaan. Toistotietojen lähdeaineistoon tehdään uusi haku, jossa haetaan viimeisen 24 tunnin ajalta tiedot siitä, mitä ohjelmia silloin on toistettu sekä milloin kyseinen ohjelma on julkaistu, jolloin saadaan uusi toistotietojen tensori $\boldsymbol{Y}'$.

Viimeisen 24 tunnin aikana julkaistujen ohjelmien suosioiden oletetaan olevan gamma-jakautuneita

$$\pi'_i \sim \text{Gamma}(\alpha_i, \beta_i).$$

Gamma-jakaumien parametrit sovitetaan siten, että virhe suhteessa tensoriin $\boldsymbol{Y}'$ minimoidaan.

Uusi tensori

$$\boldsymbol{\Lambda}'_{uci} = \frac{\pi'_i}{\pi_i}\boldsymbol{\Lambda}_{uci} = \frac{\pi'_i}{\pi_i}\sum_k \boldsymbol{U}_{uk}\boldsymbol{C}_{ck}\boldsymbol{I}_{ik},$$

eli vanhasta tensorista $\boldsymbol{\Lambda}$ jaetaan pois entiset ohjelmien suosion tasot π_i ja kerrotaan uudet suosioiden tasot π'_i tilalle. Ohjelmien suosioiden tasot π'_i poimitaan toisistaan riippumattomasti Thompson-satunnaisotannalla. Thompson-satunnaisotanta esitellään liitteessä 3.

Ohjelmien suosittelu käyttäjälle

Kun käyttäjälle u ennustetaan, millä todennäköisyydellä ohjelmasarja i kiinnostaisi häntä käyttökontekstissa c, se saadaan laskettua Poisson-parametrien odotusarvojen mukaisesti

$$\text{tulos}_{uci} = E[\boldsymbol{\Lambda}_{uci}].$$

Kun näin toimitaan kaikkien ohjelmasarjojen osalta käyttäjälle u käyttökontekstissa c, saadaan vektori tulos_{uc}, jonka avulla ohjelmasarjat voidaan järjestää käyttäjälle kiinnostavuusjärjestykseen. Toisin sanoen, toistamattomat sisällöt järjestetään todennäköisyyden mukaan, mitä sisältöä käyttäjä toistaisi todennäköisimmin.

Samankaltaisten ohjelmien suosittelu

Mikäli käyttäjälle u suositellaan jonkin ohjelman i kanssa samankaltaisia ohjelmia, tällöin 60-komponenttisesta ohjelmamatriisista $\boldsymbol{I}$ haetaan toisia ohjelmia, joiden ko-

sinietäisyys ohjelmaan i on mahdollisimman pieni, ja joita kyseinen käyttäjä u ei ole vielä nähnyt.

Suodatukset suosituksiin

Areenan suosittelun tuloksiin tehdään muutamia suodatuksia. Yle Areenassa on sekä ruotsinkielinen puoli että suomenkielinen puoli. Suositukset siis suodatetaan Areenassa sen mukaan, kumman kielisessä Areenassa suosituksia näytetään. Myös lastenohjelmat suodatetaan pois yleisen Yle Areenan puolella suositeltavista ohjelmista. Edellä mainittujen lisäksi suosituksista suodatetaan pois niin sanotut klipit, eli sellaiset mediat, joiden ei voida katsoa olevan kokonaisia ohjelmia.

Kylmäkäynnistysongelma

Kuten luvussa 3.1.3 todettiin, sekä uusi käyttäjä että uusi ohjelma on tyypillisesti ongelma yhteistoiminnallisille suosittelualgoritmeille. Kylmäkäynnistysongelma vaikuttaa Areenassa erityisen paljon, koska Areenan ohjelmisto vaihtuu koko ajan. Areenaan julkaistaan uutta sisältöä joka päivä ja vastaavasti vanhaa sisältöä poistuu päivittäin.

Yle Areenan suosittelussa uudeksi käyttäjäksi mielletään käyttäjä, joka ei ole vielä toistanut Areenasta mitään. Uusien käyttäjien suhteen kylmäkäynnistysongelma on ratkaistu siten, että uuden käyttäjän rivi matriisissa $\boldsymbol{U}$ on alussa täynnä ykkösiä, eli käyttäjä on kiinnostunut jokaisesta komponentista tai aiheesta yhtä paljon.

Kylmäkäynnistysongelma koskee myös uusia ohjelmia. Uudet ohjelmat, joita Yle Areenaan tulee koko ajan, hyötyvät Thompson-satunnaisotannasta. Ohjelman suosioita satunnaistamalla uusi ohjelma saa korkean suosion osalle käyttäjistä, ja pääsee siten suosituslistoille.

4.1.2 Tutkimusta varten tehdyt muutokset järjestelmään

Yle Areenan suosittelujärjestelmän toimintaperiaate pidettiin pitkälti ennallaan. Tein järjestelmään muutoksen, jonka myötä mallien opetustietoina toimivista toistotiedoista on mahdollista suodattaa jäljelle pelkät lastenohjelmat. Tällöin suosittelumalleja saadaan opetettua pelkkien lastenohjelmien toistotiedoilla. Toisaalta Areenan suosittelujärjestelmän rajapinnassa on jo olemassa mahdollisuus rajata suositukset pelkästään yhteen kategoriaan. Mikäli siis suosittelumalli on opetettu kaikkien ohjelmien toistotiedoilla, suosituksiin voidaan rajata jäljelle pelkät lastenohjelmat. Näin saman algoritmin avulla saadaan toteutettua kaksi järjestelmää.

Areenan suosittelujärjestelmän tuotantoversio hakee aina tuoreimmat toistotiedot, joiden pohjalta suosittelumalli tuotetaan. Tässä tutkimuksessa on kuitenkin oleellista, että vertailtavat mallit tuotetaan samoilla toistotiedoilla. Tämän vuoksi muokkasin järjestelmää siten, että ennalta määritellyt toistotiedot on mahdollista syöttää järjestelmään kullekin mallin opetuskerralle. Tällöin voitiin varmistua siitä, että mallien tuloksia vertailtaessa mahdollisten erojen syynä eivät ole eroavaisuudet mallien opetuksessa käytetyissä toistotiedoissa.

4.2 Muistipohjainen suosittelija

Tein tätä tutkielmaa varten muistipohjaisen suosittelijan luvussa 3.1.1 kuvattujen algoritmien mukaisesti. Toteutin muistipohjaiseen suosittelijaan sekä käyttäjäpohjaisen - että ohjelmapohjaisen suosittelualgoritmin. Muistipohjaisen suosittelijan lähdekoodit ovat saatavilla Bitbucketissa Eclipse Public lisenssillä [1].

Muistipohjainen suosittelija käyttää samassa muodossa olevia toistotietoja kuin Areenan suosittelujärjestelmä. Tämän vuoksi toistotietojen sisäänlukufunktioissa on otettu vaikutteita Areenan suosittelujärjestelmästä. Toistotiedoista koostetaan matriisi $\boldsymbol{Y}$, jossa on rivi käyttäjää u kohden ja sarake ohjelmasarjaa i kohden. Koska muistipohjaiset suosittelujärjestelmät perustuvat siihen, että arvioinnissa käytetään useampiportaista kuin binääristä asteikkoa, päätin käyttää käyttäjän toistamaa prosenttiosuutta ohjelmasta sen arviona. Matriisin $\boldsymbol{Y}$ solujen arvoina on siis prosenttiosuus, jonka käyttäjä u oli toistanut ohjelmasta i.

Toistotietoihin tehdään suodatuksia samoilla periaatteilla kuin Areenan suosittelujärjestelmässäkin. Toistotiedoista jätetään jäljelle pelkkien lastenohjelmien toistotiedot. Lisäksi audio-ohjelmat suodatetaan pois. Tiedoista jätetään jäljelle ne katselut, joissa ohjelmaa oli katseltu vähintään 40 %. Lisäksi puutteellisin tiedoin varustetut lähdetietorivit poistetaan. Tällaisia ovat esimerkiksi rivit, joista puuttuu käyttäjän tunniste. Mikäli käyttäjä on katsonut samaa ohjelmaa useampaan kertaan, näistä tiedoista jätetään jäljelle vain yksi katselukerta. Jotta tulokset olisivat vertailukelpoisia Areenan suosittelujärjestelmän kanssa, myös muistipohjaisessa suosittelijassa saman sarjan jaksot koostetaan yhteen, jolloin matriisin $\boldsymbol{Y}$ sarakkeet i kuvastavat yksittäisten ohjelmien sijaan kokonaisia ohjelmasarjoja. Matriisin $\boldsymbol{Y}$ solujen arvoiksi tällöin lasketaan sarjan jaksojen katseluprosenttiosuuksien keskiarvo. Järjestelmässä ohjelmasarjojen arvosanana on siis reaaliluku 0,4 ja 1 väliltä. 0,4 tarkoittaa huonoa arviota ja 1 tarkoittaa hyvää arviota. Mikäli matriisin $\boldsymbol{Y}$ solun arvona on 0, tä-

[1] https://bitbucket.org/tuukkap/mem-based-recommender

mä tarkoittaa sitä, ettei käyttäjä ollut katsellut kyseistä ohjelmasarjaa. Järjestelmä siis pyrkii ennustamaan, mikä käyttäjän tuleva katseluprosentti olisi ohjelmasarjalle, jota hän ei ole vielä nähnyt.

4.2.1 Käyttäjäpohjainen suosittelu

Toteutin muistipohjaiseen suosittelijaan käyttäjäpohjaisen suosittelun kahdella algoritmilla. Toteutin luvussa 3.1.1 kuvatun käyttäjäpohjaisen algoritmin, jota tässä nimitetään käyttäjäpohjaiseksi algoritmiksi 1. Toteutin myös toisen käyttäjäpohjaisen suosittelualgoritmin mukaillen ohjelmapohjaisen suosittelun algoritmia, joka kuvataan luvuissa 3.1.1 ja 4.2.2. Tämä algoritmi nimettiin käyttäjäpohjaiseksi algoritmiksi 2. Kummassakin algoritmissa käyttäjien samankaltaisuutta mitataan Pearson-korrelaation avulla.

Järjestelmä laskee ennusteet käyttäjää kohden koko ohjelmasarjavektorille kerralla. Laskutoimitukset toteutetaan matriisilaskutoimitusten avulla. Korrelaatiofunktio

$$\boldsymbol{W}_{uu'} = \frac{\sum_i (\boldsymbol{Y}_{ui} - \overline{Y}_u)(\boldsymbol{Y}_{u'i} - \overline{Y}_{u'})}{\sqrt{\sum_i (\boldsymbol{Y}_{ui} - \overline{Y}_u)^2 \sum_i (\boldsymbol{Y}_{u'i} - \overline{Y}_{u'})^2}}$$

on muutettu muotoon

$$\boldsymbol{W}_u = \frac{(\boldsymbol{Y}_u - \overline{\boldsymbol{Y}}_u)(\boldsymbol{Y} - \overline{\boldsymbol{Y}})^\intercal}{\sqrt{(\boldsymbol{Y}_u - \overline{\boldsymbol{Y}}_u)^{\circ 2}((\boldsymbol{Y} - \overline{\boldsymbol{Y}})^{\circ 2})^\intercal}}.$$

Tuloksena saadaan yksirivinen matriisi $\boldsymbol{W}_u$, jonka arvoina on käyttäjän u korrelaatiot kaikkiin muihin käyttäjiin nähden. $\overline{\boldsymbol{Y}}$ on matriisi, jossa on kullakin rivillä u kyseisen käyttäjän toistoprosenttiosuuksien keskiarvo samoissa soluissa, joissa käyttäjällä u on toistomerkintöjä matriisissa $\boldsymbol{Y}$. Eli jos ja vain jos $\boldsymbol{Y}_{ui} > 0$ niin $\overline{\boldsymbol{Y}}_{ui} > 0$. Näin toimitaan siitä syystä, että erotuksella $\boldsymbol{Y} - \overline{\boldsymbol{Y}}$ saadaan matriisi, jota voidaan käyttää tarvittavien matriisikertolaskujen tekemiseen. Yllä käytetään merkintää $\boldsymbol{X}^{\circ 2}$, joka tarkoittaa matriisin $\boldsymbol{X}$ jokaisen alkion korottamista toiseen potenssiin. Tällä tavoin käyttäjän u korrelaatiot muihin käyttäjiin nähden lasketaan kerralla.

Ennusteita laskeva funktio

$$\boldsymbol{P}_{ui} = \overline{\boldsymbol{Y}}_u + K_u \sum_{u'} \boldsymbol{W}_{uu'}(\boldsymbol{Y}_{u'i} - \overline{\boldsymbol{Y}}_{u'})$$

on muokattu alla olevaan muotoon **käyttäjäpohjaiseksi algoritmiksi 1**.

$$\boldsymbol{P}_u = \overline{Y}_u + K_u(\boldsymbol{W}_u(\boldsymbol{Y} - \overline{\boldsymbol{Y}})^\intercal).$$

Ennusteet kaikista ohjelmista lasketaan siis kerralla käyttäjälle u. Yksirivisessä matriisissa $\boldsymbol{W}_u$ asetetaan käyttäjän u oman solun arvoksi 0, jotta myös käyttäjän jo katsomille ohjelmasarjoille laskettaisiin uusi ennuste, jonka tarkoituksena on kuvastaa sitä, millä todennäköisyydellä käyttäjä u katsoisi ohjelmasarjaa i uudelleen. Kertoimeksi K lasketaan yksi jaettuna korrelaatioiden itseisarvojen summalla, eli

$$K_u = \frac{1}{\sum_{u'} |\boldsymbol{W}_{uu'}|}.$$

Lopputuloksena on yksirivinen matriisi $\boldsymbol{P}_u$, jossa on ennuste kustakin ohjelmasarjasta käyttäjälle u, minkälaisella toistoprosenttiosuudella hän saattaisi katsella kutakin ohjelmaa.

Kokeilin myös vaihtoehtoista funktiota, joka oli mukailtu luvussa 3.1.1 kuvatusta ohjelmapohjaisesta funktiosta. **Käyttäjäpohjainen algoritmi 2** laskee ennusteen ohjelmasarjasta i käyttäjälle u siten, että

$$\boldsymbol{P}_{ui} = \frac{\sum_{u'} \boldsymbol{W}_{uu'} \boldsymbol{Y}_{u'i}}{\sum_{u'} |\boldsymbol{W}_{uu'}|}.$$

4.2.2 Ohjelmapohjainen suosittelu

Käyttäjäpohjaisen suosittelun tapaan ohjelmapohjainen suosittelu noudattaa luvussa 3.1.1 kuvattuja algoritmeja. Myös ohjelmapohjaisessa suosittelussa ohjelmien väliset korrelaatiot lasketaan Pearson-korrelaation avulla, jossa funktio on muutettu matriisilaskutoimituksiksi eli

$$\boldsymbol{W}_i = \frac{(\boldsymbol{Y}_i - \overline{\boldsymbol{Y}}_i)(\boldsymbol{Y} - \overline{\boldsymbol{Y}})^\intercal}{\sqrt{(\boldsymbol{Y}_i - \overline{\boldsymbol{Y}}_i)^{\circ 2}((\boldsymbol{Y} - \overline{\boldsymbol{Y}})^{\circ 2})^\intercal}}.$$

Koska samoja ohjelmakorrelaatiovektoreita saatetaan käyttää useamman käyttäjän suositteluja laskettaessa, tein sovellukseen yksinkertaisen välimuistin, joka taltioi käytettyjä ohjelmakorrelaatiovektoreita, jottei jo laskettua vektoria tarvitsisi laskea uudelleen yllä olevalla kaavalla.

Muutin ennusteen laskevan funktion myös matriisilaskutoimituksiksi muotoon

$$\boldsymbol{P}_u = \frac{\boldsymbol{W'}\boldsymbol{Y_u}^\intercal}{\sum_{i'} |\boldsymbol{W'}_{ii'}|},$$

jossa $\boldsymbol{W'}$ on matriisi, johon on laskettu ohjelmasarjojen väliset korrelaatiot vain niihin sarakkeisiin, mitä ohjelmasarjoja käyttäjä u on katsonut. Nimittäjässä otetaan summa matriisista $\boldsymbol{W'}$ riveittäin. Lopputuloksena on vektori, joka koostuu ohjelmasarjojen katseluprosenttien ennusteista käyttäjälle u.

Rajapinta

Toteutin muistipohjaiseen suosittelijaan Areenan suosittelujärjestelmän tapaan rajapinnan, josta käyttäjäkohtaisia suosituksia voi kysellä. Rajapinnalle voi antaa parametriksi käyttäjän tunnisteen, käyttäjän tyypin (onko hän käyttänyt Areenaa selaimella vai mobiililaitteella), suositusten lukumäärän sekä käytettävän suosittelutavan. Rajapintaa kutsuttaessa suosittelutapavaihtoehtoina ovat muistipohjainen suosittelu, satunnainen suosittelu ja suosituimpia ohjelmia suositteleva suosittelu. Muistipohjaisen suosittelun algoritmi asetetaan koko suosittelusovellusta käynnistettäessä.

Aikavaativuus ja laskenta-aika

Kuten luvussa 3.1.1 todettiin, sekä käyttäjäpohjaisen että ohjelmapohjaisen algoritmin aikavaativuus ennusteen $\boldsymbol{P}_{ui}$ laskemiseen on $\mathcal{O}(nm)$, jossa n tarkoittaa käyttäjien määrää ja m ohjelmasarjojen määrää. Kun käyttäjälle u lasketaan ennusteet kaikista ohjelmasarjoista i, tällöin aikavaativuudeksi tulee $\mathcal{O}(nm^2)$. Koska tässä sovelluksessa tehdään kaikki suosittelut yhdelle käyttäjälle kerralla matriisikertolaskujen avulla, aikavaativuus paranee hieman. Kun matriisikertolasku suoritetaan koulukirjojen mukaisesti, aikavaativuus on $\mathcal{O}(n^3)$. Matriisikertolaskusta on kuitenkin olemassa optimoituja tapoja, joissa aikavaativuus on parhaimmillaan $\mathcal{O}(n^{2,373})$.

Sovellusta käynnistettäessä tietojen lataaminen tietokoneen muistiin kestää noin 7 minuuttia. Tämän jälkeen toistotietojen käsittely kestää noin 30 sekuntia. Kun sovellus on käynnissä, sovellus tarjoaa yhdelle käyttäjälle 20 suositusta käyttäjäpohjaisilla algoritmeilla 1 ja 2 noin 0,25 sekunnissa ja ohjelmapohjaisella algoritmilla keskimäärin 0,06 sekunnissa. Ohjelmapohjainen algoritmi hyötyy siitä, että samoja ohjelmien korrelaatiovektoreita pystytään käyttämään useampien käyttäjien ennusteiden laskennassa, jolloin niitä ei tarvitse laskea uudelleen, vaan jo laskettuja korrelaatiovektoreita voidaan käyttää uudelleen välimuistin avulla. Kun sovellusta ajettiin käytännössä ja käytettävä aineisto sisälsi noin 500 000 käyttäjää ja 1600 ohjelmasarjaa, koko aineiston läpikäyntiin eli suositusten tekemiseen kaikille käyttäjille kului aikaa käyttäjäpohjaisella algoritmilla noin 22 tuntia ja ohjelmapohjaisella algoritmilla noin 5 tuntia.

5 Tutkimuksen teko

Tässä luvussa käydään läpi tutkimuksessa käytettävät lähdetiedot, tutkimusmenetelmä sekä tulokset. Luvussa 5.1 esitellään tutkittavat järjestelmien konfiguraatiot. Opetus- ja testijakson toistotietoihin syvennytään luvussa 5.2. Valitut suosittelujärjestelmien mittaustavat esitellään luvussa 5.3. Mittaamisen toteuttaminen kuvaillaan luvussa 5.4. Lopuksi mittauksen tulokset esitellään luvussa 5.5.

Tutkimuksessa vertaillaan lasten tv-sarjojen suosittelua Areenan suosittelujärjestelmän tuottamana sekä muistipohjaisen suosittelijan tuottamana. Vertailu tehdään tuottamalla suosittelumalli opetusjakson toistotiedoilla, jonka jälkeen mallin suosittelu mitataan valituilla mittareilla testijakson toistotietoja vasten. Suosituksia ei näytetä käyttäjälle, joten testi on niin sanottu offline-testi. Jotta Yle Areenan suosittelujärjestelmän ja muistipohjaisen suosittelujärjestelmän tulokset olisivat mahdollisimman vertailukelpoisia, muistipohjainen suosittelujärjestelmä on konfiguroitu toimimaan Areenan suosittelujärjestelmän kanssa samoilla periaatteilla, kuten luvussa 4.2 kuvattiin.

Tutkimuksen tavoitetilana oli rajata toistotiedot Lasten Areenan käyttötietoihin. Koska tutkimusta tehdessä ei ollut kohtuullisella työllä toteutettavissa olevaa tapaa poimia Lasten Areena -sovelluksen käyttäjiä erilleen toistotiedoista, rajaus tehtiin siten, että toistotiedoista jätettiin jäljelle Lapset-kategoriaan kuuluvien ohjelmien toistotiedot. Toisena rajauksena tutkimus rajattiin videoiden suositteluun. Yle Areenan suosittelujärjestelmä jakaa suosittelun erikseen videoihin ja audioihin, jolloin nämä toimivat kahtena toisistaan riippumattomina suositteluina. Mikäli myös audioiden suosittelua haluttaisiin tutkia, audioiden mittaus täytyisi toteuttaa erikseen. Tämän rajauksen ei katsottu olevan tutkimuksen kannalta erityisen haitallinen, sillä valtaosa Yle Areenan ja Lasten Areenan käyttötiedoista kohdistuu videoihin. Esimerkiksi luvun 1 lopussa esitellyt suosituimmat lastenohjelmat ovat videoita. Näillä perusteilla sekä myös tutkimuksen ajanhallinnan vuoksi audiot rajattiin tästä tutkimuksesta pois. Tutkimus pystytään toistamaan tarvittaessa myöhemmin rajaten esimerkiksi tutkittavat kategoriat eri tavalla.

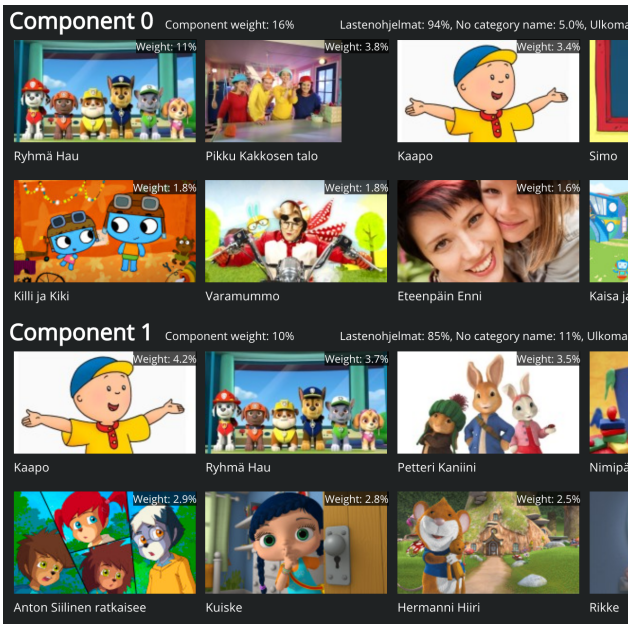
Suosittelujärjestelmien lähdetietoina toimivat Yle Areenan ohjelmien toistotiedot. Näistä suosittelujärjestelmien lähdetiedoista käytetään tässä tutkielmassa myös lyhyempää nimitystä ”toistotiedot”. Toistotiedoista ei pystytä tunnistamaan kenenkään käyttäjän henkilöllisyyttä.

Suosittelun osuvuuden mittaamiseen valittiin Areenan suosittelujärjestelmän mittauksessa käytetyt herkkyys (engl. Recall) [18] ja normalisoitu diskontattu kumulatiivinen arvo (engl. Normalized Discounted Cumulative Gain, nDCG) [21, 4]. Mo-

nimuotoisuuden mittaustavaksi valittiin käänteinen Simpson-indeksi (engl. Inverse Simpson Index) [22, 34]. Mittaustavat esitellään tarkemmin luvussa 5.3.

5.1 Tutkittavat konfiguraatiot

Yle Areenan suosittelujärjestelmän suorituskykyä lastenohjelmien suositusten tuottamisessa vertailtiin kahdella pääasiallisella konfiguraatiolla. Ensimmäinen konfiguraatio oli opettaa Areenan suosittelujärjestelmän suosittelumalli kaikkien ohjelmien toistotiedoilla, jonka jälkeen suosituksia haettaessa suosituksista suodatetaan pois kaikki muut kuin lastenohjelmat. Toinen konfiguraatio oli opettaa Areenan suosittelujärjestelmän suosittelumalli pelkkien lastenohjelmien toistotiedoilla. Jälkimmäisessä, pelkillä lastenohjelmilla opetetussa konfiguraatiossa tehtiin lisäksi kokeita muuttamalla komponenttien lukumäärää. Yle Areenan suosittelujärjestelmä käyttää oletusasetuksissa 60 komponenttia. Vertailemalla komponenttien sisältöä silmämääräisesti havaittiin, että vain lastenohjelmilla opetetussa 60 komponentin mallissa osa komponenttien sisällöistä oli sotkeutuneena keskenään. Eli toisin sanoen, komponentit eivät silmämääräisesti arvioiden näyttäneet muodostavan yhtä selkeitä kokonaisuuksia kuin kaikella lähdetiedolla opetetussa mallissa. Kuvassa 10 nähdään, miten sekä komponentti 0 että komponentti 1 sisältävät sekä Ryhmä Haun että Kaapon. Tämän vuoksi toinen konfiguraatio, eli pelkillä lastenohjelmilla opetettavat mallit, jaettiin edelleen 10, 20, 30, 40, 50 ja 60 komponenttia sisältäviin konfiguraatioihin.



Kuva 10: Pelkillä lastenohjelmilla opetetun, 60 komponenttia sisältävän mallin komponentteja.

Luvussa 4.2 esitellyssä muistipohjaisessa suosittelijassa on käytettävissä erilaisia suosittelutapoja. Suosittelijassa on kaksi vaihtoehtoista käyttäjäpohjaista suosittelualgoritmia (engl. user to user recommendation) sekä yksi ohjelmapohjainen suosittelualgoritmi (engl. item to item recommendation). Algoritmien matemaattiset taustat esiteltiin luvussa 3.1.1. Lisäksi muistipohjaisessa suosittelijassa on käytettävissä satunnaisen valikoiman ohjelmia suositteleva suosittelija sekä aina suosituimpia ohjelmia suositteleva suosittelija. Testejä ajavassa sovelluksessa on mukana myös niin sanottu ”täydellinen”, huijaava suosittelija, joka suosittelee aina niitä ohjelmia, joita käyttäjä on katsonut. Satunnainen suosittelija, suosituimpia ohjelmasarjoja suositteleva suosittelija, sekä täydellinen suosittelija otettiin mittauksiin mukaan, jotta aitojen suosittelualgoritmien suorituskykylukemat kyettiin asettamaan mittasuhteisiin. Testattavat mallit ovat nähtävissä kootusti taulukossa 3.

Taulukko 3: Testattavat mallit ja algoritmit.

Testi	**Järjestelmä**	**Lähdetiedot**	**Algoritmi tai malli**
1	Areenan suosittelujärjestelmä	Kaikki ohjelmat	60 komponenttia, suosituksissa vain lastenohjelmat
2	Areenan suosittelujärjestelmä	Lastenohjelmat	60 komponenttia
3	Areenan suosittelujärjestelmä	Lastenohjelmat	50 komponenttia
4	Areenan suosittelujärjestelmä	Lastenohjelmat	40 komponenttia
5	Areenan suosittelujärjestelmä	Lastenohjelmat	30 komponenttia
6	Areenan suosittelujärjestelmä	Lastenohjelmat	20 komponenttia
7	Areenan suosittelujärjestelmä	Lastenohjelmat	10 komponenttia
8	Muistipohjainen suosittelija	Lastenohjelmat	Käyttäjäpohjainen algoritmi 1
9	Muistipohjainen suosittelija	Lastenohjelmat	Käyttäjäpohjainen algoritmi 2
10	Muistipohjainen suosittelija	Lastenohjelmat	Ohjelmapohjainen algoritmi
11	Muistipohjainen suosittelija	Lastenohjelmat	Satunnainen
12	Muistipohjainen suosittelija	Lastenohjelmat	Suosituimmat
13	Mittaussovellus	Lastenohjelmat	”Täydellinen”

5.2 Suosittelujärjestelmien lähdetietojen tarkastelu

Kummankin tutkittavan suosittelujärjestelmän eli Yle Areenan suosittelujärjestelmän ja muistipohjaisen suosittelijan lähdetietoina toimivat Yle Areenan toistotiedot. Niistä poimittiin kaksi kokonaisuutta: opetusjakson toistotiedot ja testijakson toistotiedot.

Opetusjakson toistotiedoiksi valittiin 50 vuorokauden pituinen ajanjakso Yle Areenan yleisen suosittelun tapaan. Ajanjaksoksi valittiin lastenohjelmien katselun kannalta neutraali ajanjakso alkaen 27.8.2018 klo 13.00 ja päättyen 16.10.2018 klo 14.00. Esimerkiksi kesän tai joulun toistotietoja haluttiin välttää, koska Ylen liiketoiminnan asiantuntemuksen mukaan kummankin katsottiin olevan jossain määrin poikkeuksellisia ajanjaksoja joko ohjelmistoltaan tai lasten ajankäytön suhteen. Tätä kyseistä ajanjaksoa nimitetään opetusjaksoksi.

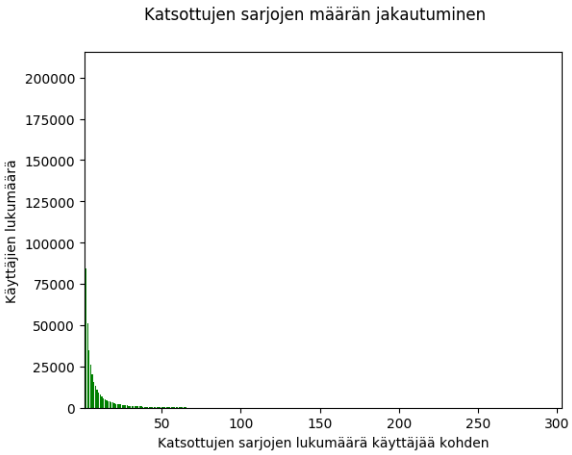
Mallit testattiin opetusjaksosta erillisiä toistotietoja vasten. Testien ajanjaksoksi valittiin opetusjaksosta välittömästi seuraavat kaksi viikkoa eli alkaen 16.10.2018 klo 14.00 ja päättyen 31.10.2018 klo 14.00. Testien ajanjaksoa nimitetään testijaksoksi. Kahden viikon ajanjakson arvioitiin olevan liiketoiminnan asiantuntemuksen mukaan riittävän pitkä ajanjakso, jonka aikana oletettu lastenohjelmien katsoja olisi tehnyt päätöksen joko katsoa suositeltua ohjelmaa tai olla katsomatta. Tietoja käyttäjien ja ohjelmasarjojen määristä opetus- ja testijaksolta on nähtävissä taulukossa 4. Ensimmäisellä rivillä on nähtävissä taulukon 3 testissä 1 käytettävät toistotiedot. Toisen rivin tiedot koskevat testejä 2-13. Kolmannella rivillä nähdään testijakson toistotietojen tietoja.

Ohjelmien toistotiedot koostuvat tietokantariveistä, joissa sarakkeina on joko selaimen tai mobiililaitteen tunniste, ohjelman tunniste, ohjelmasta toistettu millisekuntimäärä, ohjelman kesto millisekunteina sekä aikaleima, jolloin toistaminen oli aloitettu. Koska selaimen tai mobiililaitteen tunnisteet ovat omina sarakkeinaan, käyttökonteksti pystytään päättelemään tietokantarivistä sen perusteella, kumpi tunnisteista (selaimen tunniste tai mobiililaitteen tunniste) esiintyy kyseisellä tietokantarivillä.

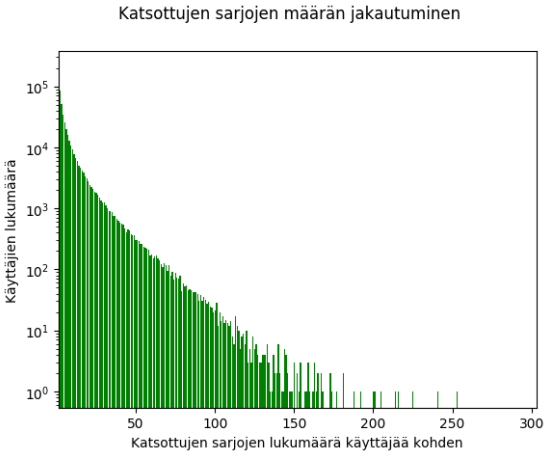
Taulukko 4: Tietoa käyttäjien ja ohjelmasarjojen määristä opetus- ja testijaksolta.

Lähdetieto	**Käytetty tieto**	**Toistokerrat sarjatasolla**	**Käyttäjien määrä**	**Sarjojen määrä**
Opetusjakso	Kaikki	13 460 922	3 617 115	44 032
Opetusjakso	Lastenohjelmat	3 309 420	553 609	1616
Testijakso	Lastenohjelmat	1 276 257	305 579	1309

Kuvissa 11 ja 12 nähdään opetusjaksolla katsottujen ohjelmasarjojen määrän jakautuminen käyttäjien kesken. Vaaka-akselin vasemmassa laidassa on vain yhden ohjelmasarjan katsoneiden käyttäjien lukumäärä. Oikeassa laidassa on useampia ohjelmasarjoja katsoneiden lukumäärä. Kuten kuvasta nähdään, Areenassa on paljon käyttäjiä, jotka ovat katsoneet vain yhtä tai muutamaa ohjelmasarjaa ja vähän käyttäjiä, jotka ovat katsoneet paljon sarjoja.

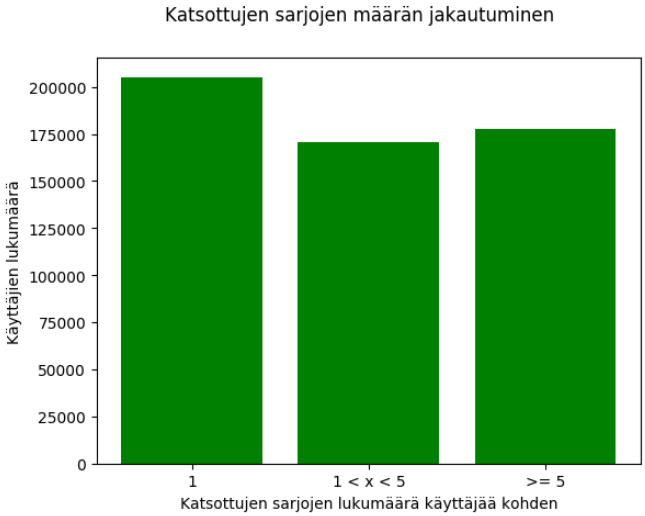


Kuva 11: Katsottujen ohjelmasarjojen määrä käyttäjää kohden.



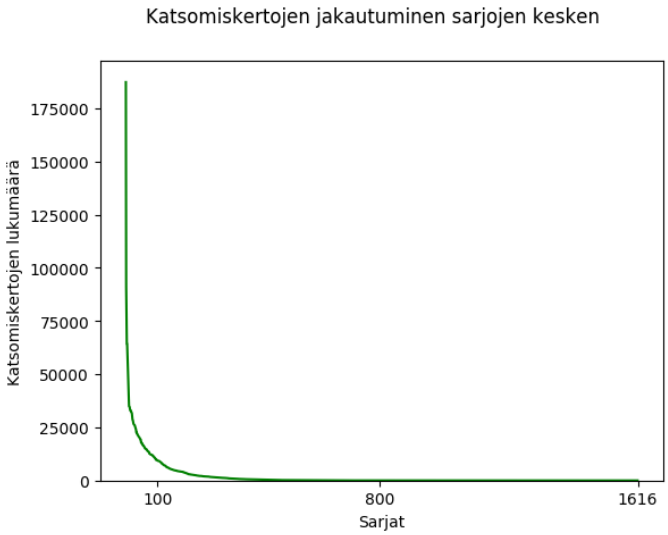
Kuva 12: Katsottujen ohjelmasarjojen määrä logaritmisella asteikolla.

Kuvasta 13 nähdään, että opetusjakson aikana 553 609 käyttäjästä 205 202 käyttäjää ovat katsoneet vain yhtä ohjelmasarjaa. Vain yhtä sarjaa katselleita on käyttäjistä siis jopa 37 %. Ohjelmasarjoja tosin käsitellään yhtenä kokonaisuutena, jolloin on mahdollista, että vain yhtä sarjaa katsoneet ovat silti voineet katsoa kyseisestä sarjasta useita jaksoja.



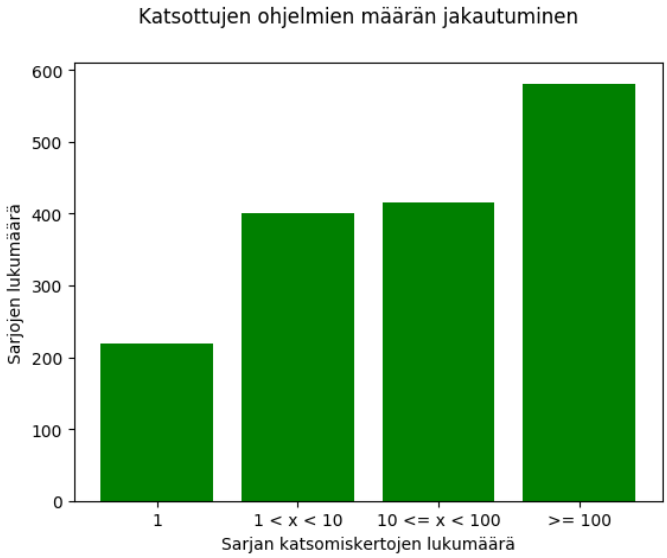
Kuva 13: Katsottujen ohjelmasarjojen määrä kategorioittain.

Myös ohjelmasarjoissa on nähtävissä voimakasta käytön keskittymistä. Kuvassa 14 nähdään johdannossa esitelty pitkä häntä -ilmiö, eli joitain harvoja ohjelmasarjoja katsotaan todella paljon, mutta suurinta osaa katsotaan hyvin vähän.



Kuva 14: Ohjelmasarjojen katsomiskertojen lukumäärä.

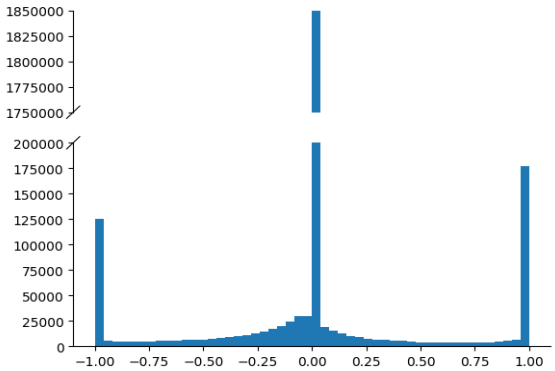
1616 ohjelmasarjasta 219 ovat sellaisia, joilla kullakin on ollut vain yksi katsoja opetusjakson aikana. Tämä on nähtävissä myös kuvasta 15. Kuten kuvasta nähdään, ohjelmasarjoista on huomattava osa myös sellaisia, joilla on enemmän kuin yksi, mutta alle 10 toistokertaa.



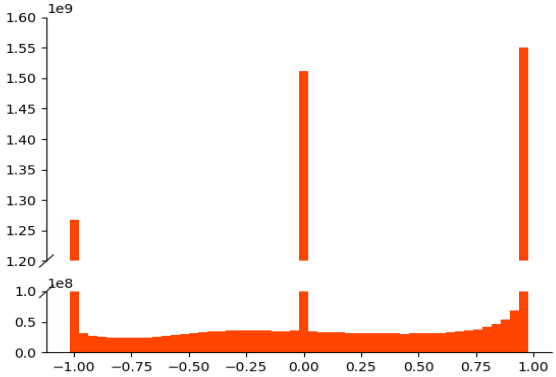
Kuva 15: Ohjelmasarjojen katsomiskertojen lukumäärä kategorioittain.

Näistä Yle Areenan toistotietojen ominaisuuksista seuraa se, että kun Areenan toistotiedot opetusjaksolta on koostettu tensoriksi $\boldsymbol{Y}$, joka sisältää myös käyttökon-

tekstit, nollasta poikkeavia arvoja on noin 0,00018 % eli tämä tensori on erittäin harva. Etenkin ohjelmien välisiä korrelaatioita on vaikea löytää, kuten havaitaan kuvasta 16. Kuvassa vaaka-akselilla on Pearson-korrelaation arvo ja pystyakselilla kunkin korrelaatioarvon lukumäärä lähdeaineiston ohjelmasarjapareista. Pearson-korrelaatio on asetettu nollaksi niissä tapauksissa, joissa Pearson-korrelaatiota ei ole määritelty. Kuvasta 17 nähdään, että käyttäjien välisiä korrelaatioita sen sijaan löytyy helpommin, joskin niidenkin arvot keskittyvät 0:aan, 1:een tai -1:een tensorin $\boldsymbol{Y}$ harvuudesta johtuen. Jos ohjelmasarjalla on vain yksi katsomiskerta, tällöin tämän asemaa muihin ohjelmiin nähden on vaikea päätellä. Sama koskee myös niitä käyttäjiä, jotka ovat katsoneet vain yhden ohjelman. Yhden ohjelman katsomiskerran perusteella on vaikea tehdä päätelmiä katsojan mieltymyksistä. Siten tämä Areenan toistotietojen ominaisuus muistuttaa kylmäkäynnistysongelmaa.



Kuva 16: Ohjelmasarjojen Pearson-korrelointi keskenään.



Kuva 17: Käyttäjien Pearson-korrelointi keskenään.

5.3 Suosittelualgoritmien mittaustavat

Tässä luvussa esitellään tutkimuksessa käytettävät suosittelualgoritmien mittaustavat. Mittareina käytetään Areenan suosittelujärjestelmien kehittämisessä käytössä olevia mittaustapoja. Herkkyys (engl. Recall) [18] ja normalisoitu diskontattu kumulatiivinen arvo (Normalized Discounted Cumulative Gain, nDCG) [21, 4] mittaavat suosittelun osuvuutta, eli sitä, missä määrin suosittelualgoritmi onnistuu tarjoamaan käyttäjälle relevanttia sisältöä. Kolmantena mittaustapana käytetään monipuolisuutta mittaavaa käänteistä Simpson-indeksiä (engl. Inverse Simpson Index) [34]. Herkkyys [18, 8] ja nDCG [4, 36] ovat suosittelualgoritmien mittaustapoina yleisiä, joten niiden käyttäminen myös tässä tutkimuksessa on perusteltua. Käänteinen Simpson-indeksi on artikkelimainintojen puutteen perusteella sisältösuosittelun parissa harvinainen. Se on kuitenkin Yleisradiolle relevantti mittari Yleisradiolaissa

määritellystä tehtävästä johtuen. Yleisradion tehtävänä ei ole ainoastaan viihdyttää yleisöä vaan tarjota myös sivistävää ja monipuolista ohjelmistoa. Tästä syystä monipuolisuuden mittaaminen on perusteltua.

5.3.1 Herkkyys- ja tarkkuus-mittarit

Jonathan Herlocker on esitellyt suosittelujärjestelmän mittaustapoja artikkelissaan ”Evaluating Collaborative Filtering Recommender Systems” [18]. Siinä hän mainitsee mittareina muun muassa herkkyyden (engl. Recall) ja tarkkuuden (engl. Precision). Herkkyys ja tarkkuus ovat hänen mukaansa suosituimpia mittaustapoja tiedonkeruujärjestelmien mittaamisessa. Herkkyys- ja tarkkuus-mittareiden lähdetiedot on kuvattu taulukossa 5. Ensimmäinen indeksi i (engl. irrelevant) tai r (engl. relevant) kertoo, sisältyvätkö arvoon käyttäjän kannalta relevantit sisältöobjektit. Toinen indeksi s (engl. seleted) tai n (engl. not selected) kuvaa sitä, valittiinko kyseinen sisältöobjekti suositeltavaksi. Toisen indeksin puuttuessa arvo sisältää kummankin kyseisen vaihtoehdon. Sana ”relevantti” tarkoittaa sisältöobjekteja, jotka käyttäjä tulisi arvioimaan eksplisiittisesti arvioimalla korkealle.

Kun käyttäjien palaute sisältöobjekteista tapahtuu eksplisiittisesti jonkinlaisella asteikolla, esimerkiksi yhdestä viiteen, tällöin tältä asteikolta valitaan jokin rajaarvo, minkä ylittävät arviot katsotaan relevanteiksi. Herlocker käytti artikkelissaan arvosana-asteikkoa yhdestä viiteen, jossa relevantteja sisältöjä olivat ne sisältöobjektit, joiden arvosana oli vähintään 3,5.

Herkkyys (engl. Recall) on

$$R = \frac{N_{rs}}{N_r}$$

eli osoittajana on niiden sisältöobjektien lukumäärä, jotka ovat käyttäjälle relevantteja ja joita on myös suositeltu käyttäjälle. Nimittäjänä on niiden sisältöobjektien lukumäärä, jotka ovat käyttäjälle relevantteja. Herkkyys kuvaa todennäköisyyttä, jolla relevantti sisältöobjekti on myös suositeltu.

Taulukko 5: Herkkyys- ja tarkkuus-mittauksessa käytettävät suureet.

	Valittu	**Ei valittu**	**Kaikki**
Relevantti	N_{rs}	N_{rn}	N_r
Ei relevantti	N_{is}	N_{in}	N_i
Kaikki	N_s	N_n	N

Tarkkuus (engl. Precision) on

$$P = \frac{N_{rs}}{N_s}$$

eli osoittajana on herkkyys-mittarin tapaan niiden sisältöobjektien lukumäärä, jotka ovat käyttäjälle relevantteja ja joita on myös suositeltu käyttäjälle. Nimittäjänä on niiden sisältöobjektien lukumäärä, joita on suositeltu käyttäjälle. Tarkkuus kuvaa todennäköisyyttä, jolla suositeltu sisältöobjekti on myös relevantti. Sekä herkkyys- että tarkkuus-arvona on reaaliluku nollan ja yhden väliltä, jossa 0 on huonoin tulos ja 1 paras tulos.

Herkkyys- ja tarkkuus-mittarit Lasten Areenassa

Herlocker [18] käyttää sanaa relevantti (engl. relevant), jolla hän tarkoittaa sitä, että käyttäjä on eksplisiittisesti ilmaissut, että kyseessä oleva sisältöobjekti miellyttää häntä. Areenassa kuitenkaan ei ole mahdollista antaa eksplisiittisiä arvioita sisällöistä, joten Areenan suositteluissa turvaudutaan implisiittiseen palautteeseen, eli siihen, onko ohjelmaa katsottu vähintään 40 % vai ei. Sen vuoksi taulukossa 5 käytetään riveillä termiä ”toistettu”. Areenan sisältöjen implisiittiset arviot ovat binäärisiä, eli ohjelmasarja joko on käyttäjälle relevantti, eli sitä on toistettu, tai sen relevanssia ei tiedetä, eli sitä ei ole toistettu. Tällöin Lasten Areenan herkkyys- ja tarkkuus-mittarien lähdetiedot ovat, kuten taulukossa 6 on esitetty.

Tarkkuus-mittarissa termi N_s toimii osamäärän jakajana. Tämä osoittautui Areenan suhteen hankalaksi. Areenan suosittelujärjestelmä järjestää kaikki tietokannassa olevat ohjelmat käyttäjäkohtaiseen paremmuusjärjestykseen. Näistä esitetään käyttäjälle n kappaletta eniten suositeltuja ohjelmasarjoja. Ei ole siis selvä asia, mitkä näistä ovat suositeltuja ja mitkä eivät. Mikäli Areenassa olisi käytössä ohjelmien arviointiasteikko, esimerkiksi yhdestä viiteen tähteä, tällöin voitaisiin määritellä raja esimerkiksi siten, että suositelluiksi katsotaan ohjelmat, joille on ennustettu arvosanaksi vähintään 3,5 tähteä. Areenassa suosittelun lopputuloksena on kuitenkin vain lista ohjelmia, jotka on järjestetty laskevaan, käyttäjäkohtaiseen paremmuus-

Taulukko 6: Herkkyys- ja tarkkuus-mittauksessa Lasten Areenassa käytettävät suureet.

	Suositeltu	**Ei suositeltu**	**Kaikki**
Toistettu	N_{rs}	N_{rn}	N_r
Ei toistettu	N_{is}	N_{in}	N_i
Kaikki	N_s	N_n	N

järjestykseen. Tällöin ei ole selvää, mikä osuus suosituksista on tarkkuus-mittarissa käytetyssä termissä N_s tarkoitettuja suosituksia. Koska Areenan suosittelujärjestelmä ennustaa todennäköisyyttä, jolla käyttäjä saattaisi valita suositellun ohjelmasarjan toistettavakseen, tämä raja periaatteessa voitaisiin etsiä todennäköisyysasteikolta nollan ja yhden väliltä. Areenan suosittelujärjestelmän rajapinta ei kuitenkaan tarjoa prosenttiosuuksia, millä todennäköisyydellä suositeltu sisältö olisi käyttäjälle relevantti. Lisäksi tässä yhteydessä saatavaan hyötyyn verrattuna kyseisen ominaisuuden lisääminen Areenan suosittelujärjestelmään ei olisi ollut järkevää. Näistä syistä tästä ajatuksesta luovuttiin. Ylen liiketoiminta on määritellyt, että tutkimuksen kohteena olevassa Lasten Areenassa 20 on lähtökohtainen lukema, minkä verran suosituksia tarjotaan käyttäjälle kerralla. Sen vuoksi päätettiin, että termi N_s on 20 eniten suositellun ohjelmasarjan joukko.

Mittarit toteutettiin siten, että mikäli käyttäjälle suositellaan juuri niitä ohjelmia, joita hän on toistanut, tällöin sekä herkkyys että tarkkuus ovat tasan 1. Näiden suhteen ilmeni tiettyjä tapauksia, joissa kumpikin mittareista antoi vääriä tuloksia. Ilmeni, että mikäli käyttäjä on toistanut alle 20 ohjelmasarjaa, täydelliselläkin suosittelujärjestelmällä tarkkuus olisi alle 1, sillä suositeltuja ja toistettuja ohjelmasarjoja on vähemmän kuin suositeltuja ohjelmasarjoja.

Otetaan esimerkki. Oletetaan, että käyttäjä u on toistanut 5 ohjelmasarjaa. Hänelle suositellaan 20 ohjelmasarjaa eli $N_s = 20$ ja suositusten joukossa ovat käyttäjän u katsomat 5 ohjelmasarjaa eli $N_{rs} = 5$. Tällöin

$$P = \frac{N_{rs}}{N_s} = \frac{5}{20} < 1.$$

Mikäli taas käyttäjä on toistanut yli 20 ohjelmasarjaa, tällöin herkkyys olisi alle 1 samasta syystä, eli suositeltuja ja toistettuja ohjelmasarjoja on vähemmän kuin suositeltuja ohjelmasarjoja.

Otetaan tähänkin esimerkki. Oletetaan, että käyttäjä u' on toistanut 24 ohjelmasarjaa eli $N_r = 24$. Hänelle suositellaan 20 ohjelmasarjaa ja suositukset ovat otos käyttäjän toistamista 24 ohjelmasarjasta eli $N_{rs} = 20$. Tällöin

$$R = \frac{N_{rs}}{N_r} = \frac{20}{24} < 1.$$

Tämän vuoksi tarkkuus-mittarin kaavaa muokattiin siten, että mikäli käyttäjä on toistanut alle 20 ohjelmasarjaa, niin jakajaksi otetaan suositusten lukumäärästä ja käyttäjän toistamien ohjelmasarjojen lukumäärästä pienempi. Herkkyys-mittarin kaavaa muokattiin samaan tapaan siten, että jakajaksi otetaan käyttäjän toista-

mien ohjelmien lukumäärästä ja suositusten lukumäärästä pienempi. Tästä seurasi se, että herkkyys ja tarkkuus olivat itse asiassa samoja arvoja. Tätä yhtä jäljelle jäävää mittaria päätettiin nimittää herkkyys-mittariksi

$$R = \frac{N_{rs}}{min\{N_r, N_s\}}.$$

5.3.2 Normalisoitu diskontattu kumulatiivinen arvo

Toiseksi mittariksi valittiin normalisoitu diskontattu kumulatiivinen arvo (engl. Normalized Discounted Cumulative Gain, nDCG) [21, 4]. Tämän mittarin merkittävin ero herkkyys-mittariin verrattuna on se, että siinä otetaan huomioon suosituksissa suositeltavien sisältöobjektien sijainti suositusten listalla. Mikäli käyttäjä on antanut korkean arvosanan suosituslistan alkupäässä olevalle sisältöobjektille, tällöin sillä on suuri merkitys. Mikäli taas korkea arvosana on kohdistunut suosituslistan häntäpäässä olevalle sisältöobjektille, tällöin sillä on pienempi merkitys. Intuitiivisesti nDCG kuvaa sitä, kuinka relevantteja sisältöjä suositellaan, ja löytyvätkö käyttäjälle relevanteimmat sisällöt suosituslistan kärkipäästä.

Oletetaan, että $\boldsymbol{P}_u$ on vektori, joka sisältää k kappaletta käyttäjälle u suositeltavia sisältöobjekteja ennustetun käyttäjäarvion mukaan laskevaan järjestykseen järjestettynä. Oletetaan myös, että $rel(\boldsymbol{P}_u, i)$ on käyttäjän u todellinen arvio suositusvektorin $\boldsymbol{P}_u$ kohdassa i olevalle sisältöobjektille. Tällöin diskontattu kumulatiivinen arvo

$$\mathrm{DCG}^{\mathrm{u}}_{\mathrm{k}} = \sum_{i=1}^{k} \frac{rel(\boldsymbol{P}_u, i)}{\log_2(i+1)}.$$

Normalisoitu diskontattu kumulatiivinen arvo lasketaan kaavalla

$$\mathrm{nDCG}^{\mathrm{u}}_{\mathrm{k}} = \frac{DCG^u_i}{IDCG_i},$$

jossa $IDCG_k$ on käyttäjälle u paras mahdollinen diskontattu kumulatiivinen arvo, joka saadaan järjestämällä suositukset $\boldsymbol{P}_u$ siten, että käyttäjän u parhaiten arvioimat ohjelmat ovat listan rel' kärjessä. Tällöin

$$\mathrm{IDCG}^{\mathrm{u}}_{\mathrm{k}} = \sum_{i=1}^{k} \frac{2^{rel'(\boldsymbol{P}_u, i)} - 1}{\log_2(i+1)},$$

jossa $rel'(\boldsymbol{P}_u, i)$ on k kappaletta käsittävä lista sisältöobjektien arvioita järjestettynä siten, että relevanteimmat sisältöobjektit ovat listan kärjessä. Normalisoidun

diskontatun kumulatiivisen arvon arvona on reaaliluku nollan ja yhden väliltä, jossa 0 on huonoin tulos ja 1 paras tulos.

nDCG Lasten Areenassa

Koska Areenassa on käytössä implisiittinen sisällön arviointi, arviointiasteikko on binäärinen eli 0 tai 1. Tällöin vektorin rel arvoina on joko 0 tai 1 sen mukaan, onko käyttäjä toistanut suositusvektorin $\boldsymbol{P}_u$ kohdassa i olevaa ohjelmaa vähintään 40 % vai ei. Vaikka implisiittisen käyttäjäarvion arvoasteikko onkin binäärinen, silti nDCG mittaa suosituksen sijoituksen oikeellisuutta, joskaan ei sitä, löytyvätkö korkeimmalle arvioidut sisältöobjektit suositusten kärkipäästä johtuen arvoasteikon binäärisyydestä.

5.3.3 Käänteinen Simpson-indeksi

Suomen laissa sanotaan Yleisradio Oy:stä näin: "Yhtiön tehtävänä on tuoda monipuolinen ja kattava julkisen palvelun televisio- ja radio-ohjelmisto siihen liittyvine oheis- ja lisäpalveluineen jokaisen saataville yhtäläisin ehdoin."(3 luku, 7 §) [1]. Laissa korostetaan myös sivistys- ja tasa-arvonäkökohtien huomioimista, mahdollisuutta oppimiseen, ja suomalaisen kulttuuriperinnön, suvaitsevaisuuden ja yhdenvertaisuuden huomioimista. Laissa tähdennetään siis sitä, että tarjonnan tulee olla monipuolista, kehittävää ja sivistävää. Luvussa 2.2 kuvaillut Yleisradion lastenohjelmiston periaatteet heijastelevat lain henkeä.

Tästä voidaan tehdä johtopäätös, että ainoa tähtäin Ylen suosittelussa ei saa olla Yle Areenan sisältöjen toistominuuttien maksimoiminen. Tärkeä näkökohta on myös ohjelmatarjonnan monipuolisuuden korostaminen. Sen vuoksi herkkyys- ja tarkkuusmittarien lisäksi otettiin käyttöön käänteinen Simpson-indeksi mittaamaan suositeltavan tarjonnan monipuolisuutta. Mittaria on käytetty Yle Areenan suosittelun mittaamisessa aiemminkin hyvin tuloksin, joten sen käyttö oli perusteltua myös tässä tutkimuksessa.

Edward H. Simpson esitteli oman nimensä saaneen, monimuotoisuutta mittaavan suureen Nature-lehdessä jo vuonna 1949 [34]. Simpson-indeksi mittaa keskittymisen tai monimuotoisuuden määrää, kun populaation yksilöt on luokiteltu ryhmiin.

Oletetaan, että äärettömässä populaatiossa kukin yksilö kuuluu johonkin R:stä ryhmästä. Oletetaan myös, että $\{p_i, p_2, ..., p_R\}, \sum_{i=1}^{R} p_i = 1$ kuvaa yksilöiden osuutta ryhmissä. Tällöin Simpsonin mukaan monimuotoisuutta mittaava arvo λ on

$$\lambda = \sum_{i=1}^{R} p_i^2.$$

Oletetaan nyt, että N yksilöä valitaan populaatiosta satunnaisesti. Oletetaan myös, että $\{n_i, n_2, ..., n_R\}, \sum_{i=1}^{R} p_i = N$ on eri ryhmiin kuuluvien yksilöiden lukumäärä. Tällöin voidaan osoittaa, että

$$\ell = \frac{\sum_{i=1}^{R} n_i(n_i - 1)}{N(N-1)}$$

on harhaton estimaattori λ:lle.

Simpson-indeksi saa arvokseen reaaliluvun 0 ja 1 välitä. Reaaliluvun voidaan ajatella kuvaavan todennäköisyyttä, jolla kaksi populaatiosta satunnaisesti valittua yksilöä kuuluvat eri ryhmiin.

Käänteinen Simpson-indeksi [22, 34] on

$$\frac{1}{\lambda} = \frac{1}{\sum_{i=1}^{R} p_i^2}.$$

Kun arvosta λ otetaan käänteisluku, tuloksen $\frac{1}{\lambda}$ voidaan ajatella kuvastavan ryhmien lukumäärää populaatiossa.

Monimuotoisuuden mittaaminen Lasten Areenassa

Vaikka monimuotoisuuden mittaamisen juuret ovat biologiassa [22], se soveltuu yllättävän hyvin myös suositusten monimuotoisuuden mittaamiseen, sillä laskennassa tarvittavat käsitteet löytyvät suositusten kontekstista vaivatta.

Lasten Areenan suosittelun monimuotoisuuden mittaamisessa monimuotoisuudella tarkoitetaan sitä, kuinka laaja joukko eri ohjelmasarjoja kaikista suositeltavista ohjelmasarjoista yhteensä muodostuu. Eli kun kaikille käyttäjille suositellaan ohjelmasarjoja, kuinka monta uniikkia ohjelmasarjaa tästä kaikkien käyttäjien suositusten joukosta löytyy.

Käänteisessä Simpson-indeksissä käsitteitä ovat ryhmä ja yksilö. Lasten Areenan suosituksissa ryhmänä toimii yksi ohjelmasarja ja tämän ryhmän yksilöitä ovat yksittäiset ohjelmasarjan suosittelukerrat. Lasten Areenan monimuotoisuuden mittaamisessa käytetään käänteistä Simpson-indeksiä, jossa estimoidaan λ:aa ℓ:llä, jolloin käänteinen Simpson-indeksi on

$$\frac{1}{\ell} = \frac{1}{\frac{\sum_{i=1}^{R} n_i(n_i-1)}{N(N-1)}}.$$

Esitellään seuraavaksi, miten saadaan käänteisestä Simpson-indeksistä pienin ja suurin indeksi. Otetaan ensimmäiseksi pienin indeksi. Oletetaan, että suosittelujärjestelmä suosittelee kaikille käyttäjille vain yhtä sarjaa. Suositeltavissa olevia ohjelmasarjoja eli ryhmiä on yhteensä 1616, eli $R = 1616$. Opetusjakson toistotiedoissa on käyttäjiä 553 609 kappaletta. Koska kullekin käyttäjälle annetaan 20 suositusta, tällöin yksilöiden eli suosittelukertojen määrä on $20 \times 553\ 609 = 11\ 072\ 180$. Koska järjestelmä suosittelee vain yhtä sarjaa kaikille käyttäjille, kaikki yksilöt kuuluvat yhteen ryhmään eli $n_i = 11\ 072\ 180$. Alaindeksillä i tarkoitetaan sitä sarjaa, jota on suositeltu kaikille käyttäjille. Tällöin

$$\frac{1}{\ell} = \frac{1}{\frac{11072180 \times (11072180-1)}{11072180 \times (11072180-1)}} = 1.$$

Käsitellään seuraavaksi suurin indeksi. Oletetaan nyt, että suosittelujärjestelmä suosittelee käyttäjille tasaisesti eri sarjoja. Tällöin 1616 sarjaa ovat jakautuneet tasaisesti 11 072 180 suosittelukertojen kesken eli kukin 1616 ohjelmasarjasta on saanut noin 6851 suosittelukertaa. Tällöin

$$\frac{1}{\ell} = \frac{1}{\frac{1616 \times (6851 \times (6851-1))}{11072180 \times (11072180-1)}} \approx 1616,$$

eli havaitaan, että käänteisen Simpson-indeksin arvoksi muodostui melko tarkasti ryhmien eli uniikkien ohjelmasarjojen lukumäärä.

5.4 Mittaamisen toteuttaminen

Areenan suosittelujärjestelmä koostuu taustajärjestelmästä, joka laskee suosittelumallin, sekä rajapinnasta, jonka kautta käyttäjäkohtaisia suosituksia kysellään. Areenan suosittelujärjestelmän taustajärjestelmän palvelimella laskettiin kukin malli vuorollaan, jonka jälkeen rajapinta asetettiin tarjoamaan suosituksia kyseisen mallin mukaisesti. Luvussa 5.1 esitetyssä taulukossa 3 rivillä 1 on kuvattuna malli, jossa lähdetietoina ovat kaikki ohjelmasarjat. Siinä muiden kuin lastenohjelmien suodatus suosituksista tehtiin siten, että rajapinnan kyselyihin lisättiin parametri, jossa sallittujen suositusten listalla (niin sanottu ”whitelist”) oli pelkkiä lastenohjelmia.

Kun suosittelumallin opetustietoina käytettiin pelkkiä lastenohjelmia, mallin opetus oli huomattavasti nopeampaa verrattuna kaikilla toistotiedoilla opetettuihin mallei-

hin. Käytettäessä 60 komponenttia ja kaikkien ohjelmien toistotietoja mallin opetukseen ja tietojen käsittelyyn kului hieman yli kolme tuntia. Sen sijaan lastenohjelmien toistotietoja käytettäessä 60 komponentin mallin opetukseen kului hieman yli yksi tunti. Kuuttakymmentä pienempää komponenttimäärää käytettäessä opetus sujui vieläkin nopeammin.

Koska Areenan suosittelujärjestelmän suositukset ovat luettavissa vain rajapinnasta, muistipohjainen suosittelusovellus toteutettiin samankaltaiseksi. Siihen luotiin rajapinta, joka tarjoaa käyttäjäkohtaiset suositukset Areenan suosittelujärjestelmän kanssa samankaltaisessa json-tiedostomuodossa. Muistipohjaisessa sovelluksessa mukana olevat satunnainen suosittelija ja suosituimpia ohjelmia suositteleva suosittelija käyttävät samaa rajapintaa. Nämä suosittelijat olivat kutsuttavissa lisäämällä suositteluraja pinnan kutsuun tietty parametri. Tällöin mittaamiseen riitti toteuttaa vain yksi sovellus, joka kyselee suositteluja mitattavasta järjestelmästä riippumatta samalla tavalla.

Toteutin mittaamista varten sovelluksen, joka laskee herkkyyden, normalisoidun diskontatun kumulatiivisen arvon sekä käänteisen Simpson-indeksin. NDCG:n toteutuksessa mukailin Areenan muita mittaussovelluksia. Mittaussovelluksen lähdekoodit ovat saatavilla Bitbucketissa Eclipse Public Lisenssillä [2]. Muistipohjaisen suosittelijan tapaan myös tässä sovelluksessa toistotietojen sisäänlukufunktioissa on otettu vaikutteita Areenan suosittelujärjestelmästä.

Kuten aiemmin mainittiin luvussa 5.2, mittaussovelluksen lähdetietoina toimivat testijakson toistotiedot eli 16.10.2018 klo 14.00 alkanut ja 31.10.2018 klo 14.00 päättynyt ajanjakso. Tämän kahden viikon ajanjakson oletettiin olevan riittävän pitkä aika, jonka aikana käyttäjä viimeistään tekee päätöksen, katsoisiko hän suositeltua ohjelmasarjaa vai ei. Testijakson toistotiedoista koostettiin tietorakenne, jossa oli rivi kutakin käyttäjää kohden sekä lista kyseisen käyttäjän toistamia ohjelmasarjoja. Mittaussovelluksessa ei huomioitu sitä, että löytyikö testijakson käyttäjä myös opetusjaksolta vai ei, sillä tarkoituksena oli mitata suosittelujärjestelmien suorituskykyä neutraalisti. Mikäli testijakson käyttäjistä olisi suodatettu jäljelle vain ne käyttäjät, jotka ovat toistaneet ohjelmia myös opetusjaksolla, tällöin kylmäkäynnistysongelman vaikutukset olisivat jääneet mittauksissa huomiotta. Koska kylmäkäynnistysongelma on oleellinen etenkin Areenan toistotiedoilla, mittaustulokset kerättiin myös niiltä käyttäjiltä, jotka eivät olleet toistaneet Areenasta mitään opetusjakson aikana.

Testit ajettiin kustakin luvussa 5.1 esitetyssä taulukossa 3 luetellusta mallista tai algoritmista. Kukin testi suoritettiin siten, että testijakson lähdetiedoista koostettu

[2] https://bitbucket.org/tuukkap/recommender-measurements

tietorakenne käytiin läpi rivi riviltä ottaen käsiteltäväksi kyseisen käyttäjän tunniste sekä lista hänen toistamiaan ohjelmasarjoja. Tämän jälkeen tehtiin kutakin testijakson käyttäjää kohden kysely testattavan suosittelujärjestelmän rajapintaan, josta saatiin tuloksena 20 suositelluimman ohjelmasarjan lista. Suositeltujen ohjelmasarjojen listan ja käyttäjän toistamien ohjelmasarjojen listan perusteella laskettiin käyttäjäkohtaisesti herkkyys-mittarin arvo (luku 5.3.1) ja normalisoitu diskontattu kumulatiivinen arvo, nDCG (luku 5.3.2). Tuloksista laskettiin kaikkien käyttäjien kesken keskiarvo. Käänteinen Simpson-indeksi (luku 5.3.3) laskettiin siten, että kaikki eri käyttäjille suositellut uniikit ohjelmasarjat koostettiin yhdeksi joukoksi, jossa ryhmänä toimi yksi ohjelmasarja ja ryhmän alkioita olivat ohjelmasarjan toistokerrat. Testiajon lopuksi tulokset kirjattiin ylös taulukkoon. Kustakin mittarista tulokset otettiin erikseen selainkäyttäjille ja mobiilikäyttäjille.

5.5 Tulokset

Mittaussovelluksen tuottamien tulosten oikeellisuus varmistettiin kahdella tavalla. Aluksi mittaussovellus testattiin kattavasti. Sen lisäksi mittaussovelluksella mitattiin satunnaisia ohjelmia suosittelevan suosittelijan tulokset, aina käyttäjän katsomia ohjelmia suosittelevan, täydellisen suosittelijan tulokset sekä suosituimpia ohjelmia suosittelevan suosittelijan tulokset. Satunnaisia ohjelmia suosittelevasta suosittelijasta hypoteesina oli, että herkkyys ja nDCG antavat pieniä lukemia ja käänteiseksi Simpson-indeksiksi muodostuu suuri luku. Täydellisen suosittelijan odotettiin antavan herkkyydestä ja nDCG:stä tulokseksi 1, kuten luvuissa 5.3.1 ja 5.3.2 esitettiin.

Satunnaisen suosittelijan tulokset olivat herkkyys- ja nDCG-mittarien osalta odotetusti varsin pieniä, ja monimuotoisuuden arvoksi muodostui opetusjaksolla olevien ohjelmasarjojen lukumäärä, joka oli 1616. Täydellinen suosittelija sai herkkyys- ja nDCG-mittarien arvoiksi odotetusti 1, ja monimuotoisuus käänteisellä Simpson-indeksillä mitattuna oli pieni. Suosituimpia ohjelmia suositteleva suosittelija sai herkkyys- ja nDCG-arvot nollasta yhteen ulottuvan asteikon puolivälistä. Koska suosituimpia ohjelmia suositteleva suosittelija antoi kaikille aina samat 20 ohjelmasarjasuositusta, tuli monimuotoisuuden arvoksi 20. Mittausten tulokset ovat nähtävillä kootusti taulukossa 7.

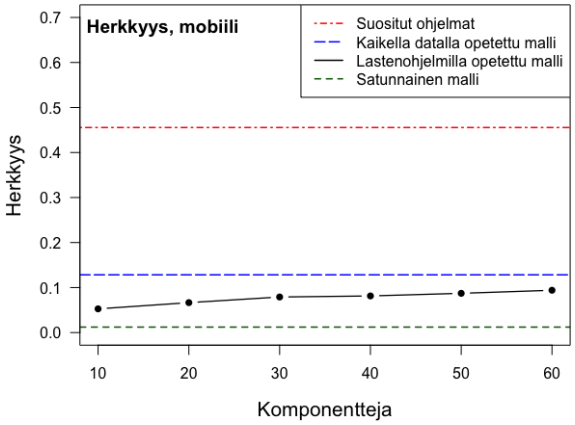
Kun Areenan suosittelujärjestelmä opetettiin lastenohjelmien toistotiedoilla, näistä malleista parhaat tulokset sai 60 komponentin malli. Sekä herkkyys- että nDCG-mittarien tulokset ylittivät satunnaisen suosittelijan tulokset selvästi. Monimuotoisuus oli sekä täydellistä suosittelijaa että suosituimpia ohjelmia suosittelevaa suosittelijaa parempi. Kun komponenttimäärää pienennettiin 10 komponentin askelin,

Taulukko 7: Suosittelumallien ja -algoritmien mittaustulokset. Ilman sulkeita olevat tulokset koskevat mobiilikäyttäjiä ja sulkeissa olevat tulokset selainkäyttäjiä.

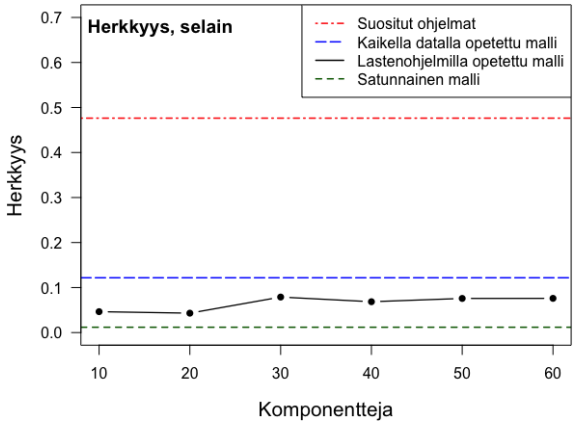
Sarjat	**Sovellus**	**Algoritmi tai malli**	**Herkkyys**	**nDCG**	**Käänt. Simpson-indeksi**
Lasten sarjat	Muistipohjainen suosittelija	Satunnainen	0,012 (0,012)	0,018 (0,014)	1616 (1616)
Lasten sarjat	Muistipohjainen suosittelija	”Täydellinen”	1 (1)	1 (1)	26 (24,26)
Lasten sarjat	Muistipohjainen suosittelija	Suositut	0,455 (0,476)	0,477 (0,469)	20 (20)
Lasten sarjat	Areenan suosittelujärjestelmä	60 komponenttia	0,094 (0,076)	0,16 (0,127)	27,62 (30,11)
Lasten sarjat	Areenan suosittelujärjestelmä	50 komponenttia	0,087 (0,076)	0,135 (0,108)	26,26 (31,93)
Lasten sarjat	Areenan suosittelujärjestelmä	40 komponenttia	0,081 (0,069)	0,128 (0,110)	26,15 (30,36)
Lasten sarjat	Areenan suosittelujärjestelmä	30 komponenttia	0,079 (0,079)	0,106 (0,104)	25,35 (33,74)
Lasten sarjat	Areenan suosittelujärjestelmä	20 komponenttia	0,066 (0,043)	0,086 (0,043)	27,93 (24,04)
Lasten sarjat	Areenan suosittelujärjestelmä	10 komponenttia	0,052 (0,046)	0,056 (0,046)	25,46 (26,14)
Kaikki sarjat	Areenan suosittelujärjestelmä	60 komponenttia, vain lasten suos.	0,128 (0,121)	0,189 (0,147)	48,23 (27,44)
Lasten sarjat	Muistipohjainen suosittelija	Käyttäjäpohjainen algoritmi 1	0,057 (0,039)	0,092 (0,064)	132,3 (231,1)
Lasten sarjat	Muistipohjainen suosittelija	Käyttäjäpohjainen algoritmi 2	0,079 (0,059)	0,120 (0,089)	125,1 (199,3)
Lasten sarjat	Muistipohjainen suosittelija	Käyttäjäpohjainen algoritmi 2 (ei jak. itseisarvoa)	0,094 (0,067)	0,134 (0,097)	90,9 (154,2)
Lasten sarjat	Muistipohjainen suosittelija	Ohjelmapohjainen	0,004 (0,006)	0,003 (0,004)	655,3 (892,2)

jokaisella askeleella sekä herkkyys- että nDCG-mittarien arvot pienenivät. Monimuotoisuudessa ei ollut havaittavissa eri komponenttien lukumäärien välillä erityisiä eroavaisuuksia.

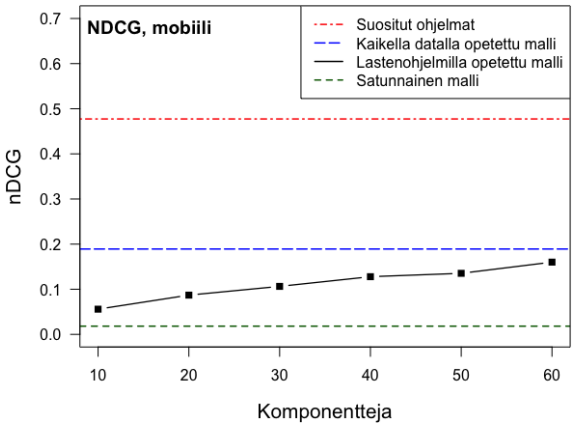
Kuvissa 18, 19, 20 ja 21 nähdään kuvaajat Areenan suosittelujärjestelmän mittausten tuloksista. Kuvissa on kuvattu yhtenäisellä mustalla viivalla eri komponenttimäärillä opetetut mallit, joissa lähdetietoina ovat lastenohjelmat. Sinisellä pitkällä katkoviivalla nähdään kaikilla ohjelmilla opetettu 60 komponentin malli, vihreällä



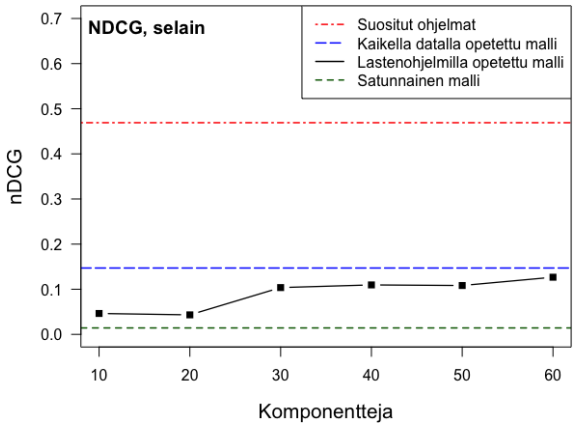
Kuva 18: Herkkyys mobiilikäyttäjiltä.



Kuva 19: Herkkyys selainkäyttäjiltä.



Kuva 20: NDCG mobiilikäyttäjiltä.



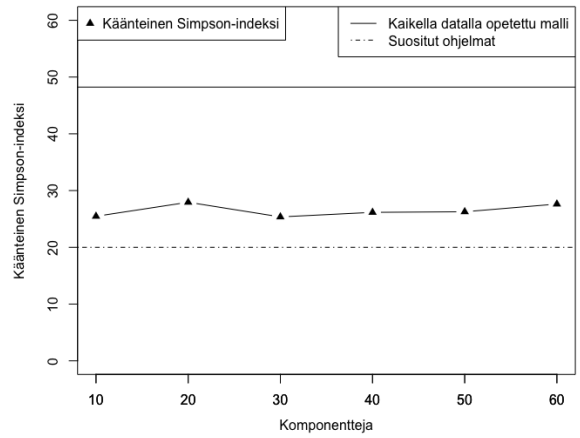
Kuva 21: NDCG selainkäyttäjiltä.

lyhyemmällä katkoviivalla nähdään satunnaisia ohjelmia suositteleva suosittelija sekä punaisella pisteellisellä katkoviivalla nähdään suosituimpia ohjelmia suositteleva suosittelija.

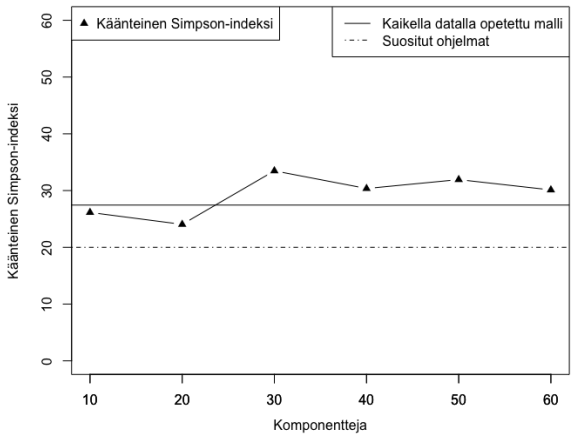
Testattavana kohteena oli myös kaikilla toistotiedoilla opetettu malli, jonka suosituksista poimittiin nähtäville vain lastenohjelmien suosittelut. Tämä malli saavutti paremmat tulokset sekä herkkyydellä, nDCG:llä että käänteisellä Simpson-indeksillä mitattuna jokaiseen lastenohjelmilla opetettuun malliin verrattuna.

Kiinnostavana havaintona mobiilikäyttäjien tulokset ovat lähestulkoon kaikissa testatuissa malleissa tai algoritmeissa selainkäyttäjiä parempia.

Kuvissa 22 ja 23 nähdään tuloksia Areenan suosittelujärjestelmän käänteisistä Simpson-indekseistä. Viivatyypit tarkoittavat malleja samalla tavoin kuin kuvissa 18, 19, 20 ja 21.



Kuva 22: Käänteinen Simpson-indeksi mobiilikäyttäjiltä.



Kuva 23: Käänteinen Simpson-indeksi selainkäyttäjiltä.

Muistipohjaisista algoritmeista käyttäjäpohjaisen algoritmi 1:n tulokset ovat verrattavissa normalisoidulla diskontatulla kumulatiivisella arvolla (nDCG) mitattuna lastenohjelmilla opetetun Areenan suosittelujärjestelmän 20 tai 30 komponentin mallin suorituskykyyn etenkin, kun seurataan mobiilikäyttäjien arvoja. Käyttäjäpohjainen algoritmi 2 sai nDCG:llä mitattuna Areenan suosittelujärjestelmän 30 tai 40 komponentin mallin kanssa samankaltaisia tuloksia. Monimuotoisuus Areenan suosittelujärjestelmään verrattuna on selvästi korkeampi siitä teknisestä syystä, että mikäli muistipohjaisen suosittelijan matriisista ei löytynyt testijakson käyttäjää, tällöin suosituksiksi annettiin 20 satunnaista ohjelmasarjaa. Kiinnostavana havaintona oli se, kun käyttäjäpohjaisen algoritmi 2:n nimittäjästä jätettiin itseisarvon laskenta pois, niin algoritmi toimi nDCG:llä mitattuna yhtä hyvin kuin lastenohjelmilla opetettu Areenan suosittelujärjestelmän malli 50 komponentilla mobiilikäyttäjien osalta. Herkkyys ylti mobiilikäyttäjillä jopa samaan kuin lastenohjelmilla opetetulla 60 komponentin mallilla. Kun käyttäjäpohjaisessa algoritmi 2:ssa ei käytetä nimittäjässä itseisarvoa, ennuste on muotoa

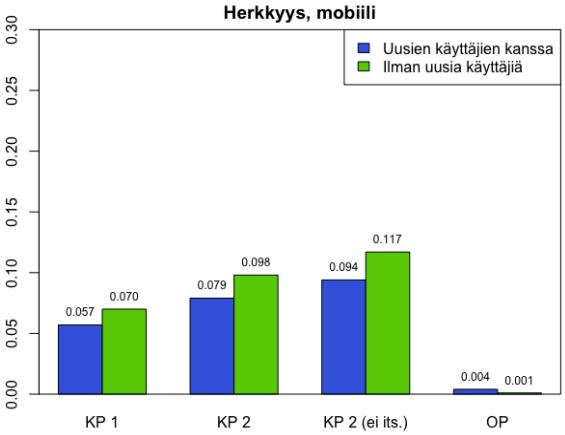
$$\boldsymbol{P}_{ui} = \frac{\sum_{u'} \boldsymbol{W}_{uu'} \boldsymbol{Y}_{u'i}}{\sum_{u'} \boldsymbol{W}_{uu'}}.$$

Ohjelmapohjainen suosittelu sen sijaan ei käytännössä toiminut satunnaista suosittelua paremmin.

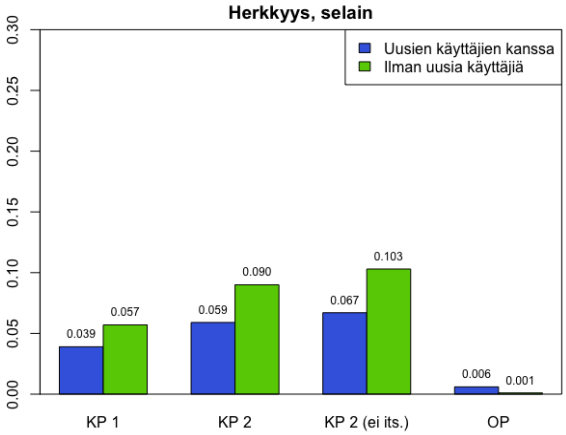
Koska muistipohjaisessa suosittelijassa uudet käyttäjät saavat satunnaiset suositukset, niiden vaikutus tuloksiin on oletettavasti huomattava. Uusilla käyttäjillä tarkoitetaan niitä testijakson käyttäjiä, joita ei löydy opetusjakson tiedoista. Siksi muistipohjaisen suosittelijan osalta tulokset mitattiin myös ilman uusia käyttäjiä eli ilman kylmäkäynnistysongelman vaikutusta. Näissä testeissä sekä herkkyys että nDCG pa-

rani hieman ja käänteinen Simpson-indeksi pieneni. Muistipohjainen suosittelija ei silti saavuttanut samanlaisia tuloksia kuin kaikkien ohjemien toistotiedoilla opetettu Areenan suosittelujärjestelmä.

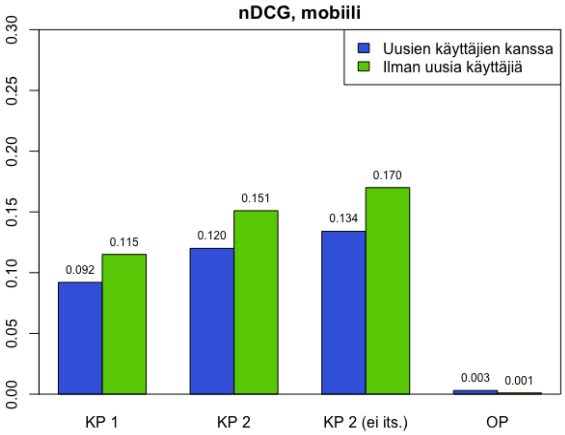
Muistipohjaisen suosittelijan mittausten tulokset ovat nähtävillä kuvissa 24, 25, 26, 27, 28 ja 29. Taulukon 7 tietojen lisäksi kuvissa on nähtävillä myös tulokset ilman uusia käyttäjiä. Kuvissa ”KP” tarkoittaa käyttäjäpohjaista algoritmia ja ”OP” tarkoittaa ohjelmapohjaista algoritmia.



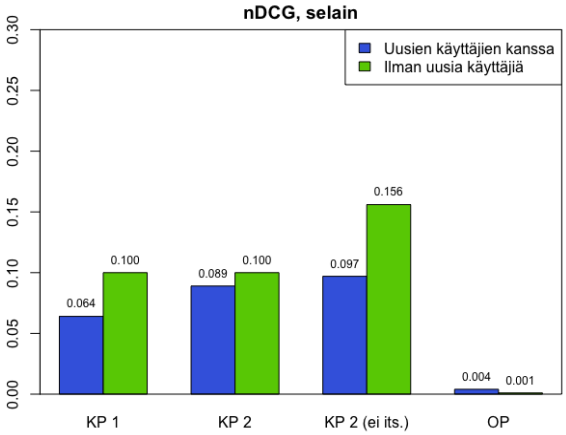
Kuva 24: Mobiilikäyttäjien herkkyys muistipohjaiselta suosittelijalta.



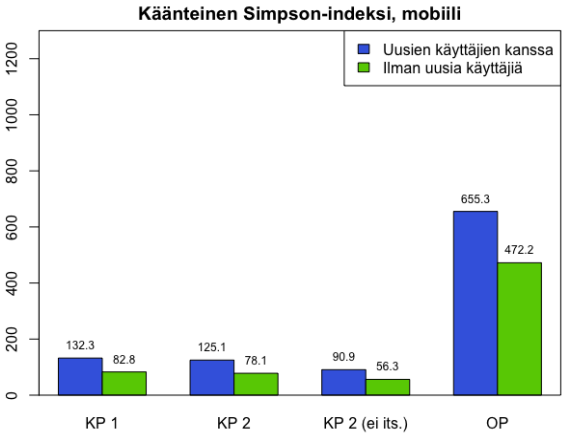
Kuva 25: Selainkäyttäjien herkkyys muistipohjaiselta suosittelijalta.



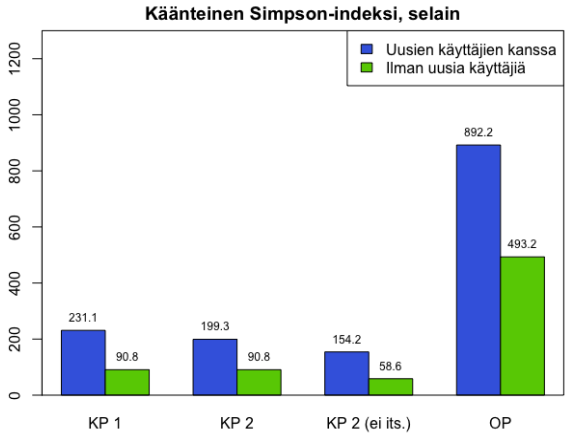
Kuva 26: Mobiilikäyttäjien nDCG muistipohjaiselta suosittelijalta.



Kuva 27: Selainkäyttäjien nDCG muistipohjaiselta suosittelijalta.



Kuva 28: Mobiilikäyttäjien käänteinen Simpson-indeksi muistipohjaiselta suosittelijalta.



Kuva 29: Selainkäyttäjien käänteinen Simpson-indeksi muistipohjaiselta suosittelijalta.

6 Pohdinta

Ennen tuloksia hypoteesinä oli, että pelkästään lastenohjelmilla opetetut Areenan suosittelujärjestelmän mallit tuottavat parhaimpia suosituksia. Koska vain osa Areenan komponenteista sisälsi lastenohjelmia, odotettiin, että parhaat tulokset tulevat 60 komponenttia pienemmällä komponenttimäärällä. Tämä perustui intuitioon siitä, että kun suosittelumalli voi kohdistaa kaiken ”energiansa” vain lastenohjelmiin, tällöin malli kykenee löytämään lastenohjelmien katsojien keskuudesta yleistä mallia paremmin korrelaatioita. Tulokset eivät kuitenkaan olleet tämän ennakko-odotuksen mukaisia.

Muistipohjaisesta suosittelijasta hypoteesiä vuorostaan oli se, että ohjelmapohjainen suosittelu tuottaa parempia tuloksia. Oletus perustui siihen, että ohjelmia keskenään vertailtaessa samankaltaisuuksia saattaa löytyä helpommin kuin käyttäjien väliltä, sillä ohjelmien voidaan olettaa olevan olioina yksinkertaisempia ja ennustettavampia kuin ihmisten. Myöskään muistipohjaisessa suosittelijassa tulokset eivät vastanneet odotuksia.

Areenan suosittelujärjestelmä

Mittaustulokset eivät kuitenkaan noudattaneet ennakko-odotuksia. Kaikilla ohjelmilla opetettu malli sai parhaat tulokset sekä suosittelun osuvuutta mittaavilla herkkyys- ja nDCG-mittareilla että monimuotoisuutta mittaavalla käänteisellä Simpson-indeksillä. Tämä oli yllättävää siitä syystä, että intuitiivisesti ajatellen suosittelun osuvuus ja monimuotoisuus vaikuttaisivat olevan keskenään ristiriitaisia tavoitteita. Jos suosittelu on hyvin osuvaa, se ei oletettavasti voi olla samaan aikaan erityisen monimuotoista. Satunnaisen suosittelijan, täydellisen suosittelijan ja suosituimpia ohjelmia suosittelevan suosittelijan tulokset tukevat tätä oletusta. Varsinaisten suosittelumallien ja -algoritmien mittaustulokset kuitenkin ovat tämän kanssa ristiriidassa.

Komponenttien lukumäärän muutoksella ei ollut odotusten mukaista vaikutusta. Odotuksena siis oli, että pienemmän komponenttimäärän avulla malli löytää korrelaatioita käyttäjien ja ohjelmien väliltä helpommin. Tulokset kuitenkin osoittivat, että mitä pienempää komponenttimäärää käytetään, sitä huonompia tuloksia saadaan. Hu, Koren ja Volinsky [19] tutkivat matriisihajotelmiin perustuvan suosittelujärjestelmän toimintaa. Myös he tekivät kokeita erilaisilla komponenttien lukumäärillä. Heidän testeissään käytettiin kymmenestä kahteensataan komponenttia. Heidän saamiensa tulosten mukaan suosittelun osuvuus parani tasaisesti, kun komponenttien

lukumäärää lisättiin. Parhaan tuloksen he saivat 200 komponentin mallilla. Tässä tutkimuksessa saadut tulokset ovat siis yhdenmukaisia heidän havaintojensa kanssa. Areenan suosittelujärjestelmässä mallin opetusaika kasvoi sitä mukaa kuin komponenttien määrää kasvatettiin. Vaikka siis teoriassa suurempi komponenttimäärä tarjoaa parempaa suosittelua, käytännössä kyse on tasapainosta käytettävissä olevien laskentaresurssien ja tarvittavan suosittelutarkkuuden välillä.

Olisi loogista, että matriisihajotelma mahdollistaa eri ikäisille lapsille sopivien suositusten tuottamisen. Mallipohjainen suosittelu perustuu siihen, että malliin taltioidaan kullekin käyttäjälle ja ohjelmalle yksilölliset komponenttivektorit, joista suositukset lasketaan. Tämän vuoksi olisi odotettavissa, että eri ikäkausille sopivat suositukset syntyvät automaattisesti. Tulosten mukaan Areenan suosittelujärjestelmä toimii muistipohjaista suosittelijaa paremmin, mikä tukee tätä arviota.

Toistotiedoissa käyttäjäkunta jakautui selainkäyttäjiin ja mobiilikäyttäjiin. Vaikka on mahdollista tai jopa todennäköistä, että joku selainkäyttäjä ja joku mobiilikäyttäjä ovat sama henkilö, tätä yhteyttä ei nykyisessä muodossa olevista toistotiedoista kyetä havaitsemaan. Kuten tuloksista nähdään, mobiilikäyttäjien tulokset ovat lähes kaikkien testattujen mallien osalta parempia selainkäyttäjiin verrattuna. Toistotiedoissa mobiilikäyttäjien määrä on selainkäyttäjiä hieman suurempi. Opetusjakson toistotiedoissa oli 229 532 yksilöllistä lastenohjelmien selainkäyttäjää ja 324 077 yksilöllistä lastenohjelmien mobiilikäyttäjää. Tällöin on mahdollista, että korrelaatioita mobiilikäyttäjien väliltä löytyy helpommin kuin selainkäyttäjien väliltä. On myös mahdollista, että Lasten Areenaa käytetään mobiilisovelluksen kautta eri tavalla kuin selaimen kautta, mikä myös osaltaan voisi selittää näitä havaintoja. Varmaa syytä tälle ilmiölle on kuitenkin vaikea löytää.

Tämän tutkielman puitteissa ei ollut keinoa tunnistaa Lasten Areenan käyttäjiä toistotietojen seasta. Sen sijaan Lasten Areenan katselutietoja estimoitiin siten, että mallien opetusmateriaalina käytettiin lastenohjelmien toistotietoja. Yle Areenassa lastenohjelmia pystyy katsomaan sekä Lasten Areenan kautta että normaalin, ”aikuisten” Areenan kautta valitsemalla Lapset-kategorian. Katselutiedoissa oli mukana siis kummankin väylän kautta syntyneet katselutiedot. Tällöin on mahdollista, että kaikilla toistotiedoilla opetettu malli on hyötynyt niiden käyttäjien osalta, jotka käyttävät myös normaalia Yle Areenaa. Kaikkien ohjelmien toistotietojen seassa on myös muiden kuin lastenohjelmien toistotietoja, jolloin järjestelmällä on enemmän tietoa tehdä päätelmiä käyttäjien ja ohjelmien välisistä yhteyksistä. Kuten luvussa 5.2 esitetystä taulukosta 4 nähdään, kaikki ohjelmasarjat sisältävissä toistotiedoissa on 3,5 miljoonaa käyttäjää ja noin 44 000 ohjelmasarjaa. Kun toistotiedoista suodatetaan jäljelle pelkkien lastenohjelmien toistot, tiedon määrä vähenee huomattavasti:

käyttäjiä on enää noin 550 000 ja ohjelmasarjoja vain 1616. Muiden kuin lastenohjelmien suodatuksen jälkeen opetusjakson toistokerrat vähenevät 13,5 miljoonasta 3,3 miljoonaan. Koneoppimisen piirissä on yleisemminkin havaittu, että mitä enemmän lähdetietoa mallin opetukseen on tarjolla, sitä pienemmällä virheprosentilla malli tyypillisesti toimii [5].

Muistipohjainen suosittelija

Muistipohjaisessa suosittelijassa käytetään ohjelmien ennusteiden $\boldsymbol{P}_{ui}$ asteikkona reaalilukua arvojen 0,4 ja 1 väliltä. Tämä toimintatapa valittiin sillä perusteella, että artikkeleissa esitellyissä funktioissa oletetaan, että ohjelmien arvioinneissa käytetään jonkinlaista useampiportaista asteikkoa. Kun otetaan huomioon, että Yle Areenassa kerätyt arviot ovat implisiittisiä, ennusteiden asteikko 0,4:stä 1:een on ongelmallinen. Siinä tehdään päätelmä, että käyttäjä ei pidä ohjelmasta, kun hän on toistanut ohjelmasta esimerkiksi vain 41 %. Voidaan kuitenkin esittää muitakin syitä sille, miksi käyttäjä on katsonut ohjelmasta keskimäärin vain tuon osuuden. Käyttäjä esimerkiksi saattaa haluta jatkaa katselua toisella kertaa, tai katselu sillä kerralla keskeytyi jonkin velvoitteen vuoksi. Prosenttiosuutta tarkastellessa saatetaan tehdä liian rohkeita johtopäätöksiä erityisesti siitä, mikä ei miellytä käyttäjää.

Myös muistipohjaisen suosittelijan mittaustulokset yllättivät. Käyttäjäpohjainen algoritmi 2 ilman jakajan itseisarvoa ylsi suorituskyvyltään jopa samalle tasolle kuin vertailun parhaimpiin kuuluvat menetelmät, kuten lastenohjelmilla opetettu Areenan suosittelujärjestelmän 50 tai 60 komponentin malli. Ei löydetty selkeää syytä sille, miksi itseisarvon pois jättäminen jakajasta paransi tuloksia.

Ohjelmapohjainen algoritmi ei tuottanut satunnaista suosittelua parempaa suosittelua, eli yksinkertaisesti ilmaisten se ei toiminut ollenkaan. Syyt ohjelmapohjaisen algoritmin huonoon suorituskykyyn löytynevät toistotiedoista. Kuten luvun 5.2 kuvasta 13 havaitaan, yli puolet käyttäjistä on katsonut ainoastaan yhtä ohjelmasarjaa.

Otetaan esimerkki. Oletetaan, että käyttäjä u on katsonut vain yhden ohjelmasarjan: i'. Kun käyttäjälle u lasketaan ennustetta jostain ohjelmasarjasta i, tällöin

$$\begin{aligned}\boldsymbol{P}_{ui} &= \frac{\sum_i \boldsymbol{W}_{ii'}\boldsymbol{Y}_{ui'}}{\sum_i |\boldsymbol{W}_{ii'}|} \\ &= \frac{\boldsymbol{W}_{ii'}\boldsymbol{Y}_{ui'}}{|\boldsymbol{W}_{ii'}|},\end{aligned}$$

jonka seurauksena

$$\boldsymbol{P}_{ui} = \begin{cases} \boldsymbol{Y}_{ui'} & \text{jos } \boldsymbol{W}_{ii'} > 0 \\ -\boldsymbol{Y}_{ui'} & \text{jos } \boldsymbol{W}_{ii'} < 0 \\ \text{Ei määritelty} & \text{jos } \boldsymbol{W}_{ii'} = 0 \end{cases}$$

jossa $\boldsymbol{Y}_{ui'}$ tarkoittaa käyttäjän u arviota ainoasta hänen katsomastaan ohjelmasarjasta i'. Niissä tapauksissa, joissa $\boldsymbol{P}_{ui}$:n arvo ei ole määritelty nollalla jakamisen vuoksi, arvoksi asetetaan 0.

Kun samalle käyttäjälle u lasketaan ennustetta toisesta ohjelmasta j, tällöin

$$\begin{aligned} \boldsymbol{P}_{uj} &= \frac{\sum_{i'} \boldsymbol{W}_{ji'} \boldsymbol{Y}_{ui'}}{\sum_{i'} |\boldsymbol{W}_{ji'}|} \\ &= \frac{\boldsymbol{W}_{ji'} \boldsymbol{Y}_{ui'}}{|\boldsymbol{W}_{ji'}|}, \end{aligned}$$

jolloin

$$\boldsymbol{P}_{uj} = \begin{cases} \boldsymbol{Y}_{ui'} & \text{jos } \boldsymbol{W}_{ji'} > 0 \\ -\boldsymbol{Y}_{ui'} & \text{jos } \boldsymbol{W}_{ji'} < 0 \\ \text{Ei määritelty} & \text{jos } \boldsymbol{W}_{ji'} = 0 \end{cases}$$

Havaitaan, että vain yhtä ohjelmasarjaa katsonut käyttäjä saa kaksi vaihtoehtoista arvoa kaikkien ohjelmien ennusteille riippuen käyttäjän katsoman ohjelman ja ennustettavan ohjelman korrelaation etumerkistä. Tällöin käyttäjän ohjelmasarjasuosituksia ei käytännössä kyetä järjestämään paremmuusjärjestykseen.

Luvussa 5.2 todettiin, että vain yhtä ohjelmasarjaa katsoneita käyttäjiä on yhteensä 205 202 kappaletta, eli yli kolmasosa kaikista 553 690 käyttäjistä. Edellä kuvattu ongelma on siis laaja. Tämän perusteella 37 % käyttäjistä ovat sellaisia, joiden suosituksia ei kyetä järjestämään.

Tarkastellaan seuraavaksi käyttäjäpohjaisen suosittelun algoritmin 2 tuottamia suosituksia. Oletetaan, että lasketaan ennustetta käyttäjälle u ohjelmasta i. Oletetaan lisäksi, että löytyy vain yksi käyttäjä u', joka on arvioinut ohjelman i. Tällöin käyttäjäpohjaisen suosittelun algoritmin 2 mukaan

$$\begin{aligned} \boldsymbol{P}_{ui} &= \frac{\sum_{u'} \boldsymbol{W}_{uu'} \boldsymbol{Y}_{u'i}}{\sum_{u'} |\boldsymbol{W}_{uu'}|} \\ &= \frac{\boldsymbol{W}_{uu'} \boldsymbol{Y}_{u'i}}{|\boldsymbol{W}_{uu'}|} \end{aligned}$$

josta seuraa

$$\boldsymbol{P}_{ui} = \begin{cases} \boldsymbol{Y}_{u'i} & \text{jos } \boldsymbol{W}_{uu'} > 0 \\ -\boldsymbol{Y}_{u'i} & \text{jos } \boldsymbol{W}_{uu'} < 0 \\ \text{Ei määritelty} & \text{jos } \boldsymbol{W}_{uu'} = 0 \end{cases}$$

Ohjelmapohjaisen algoritmin tapaan ennuste $\boldsymbol{P}_{ui}$ saa arvokseen 0, mikäli funktiossa tapahtuisi nollalla jakaminen.

Jos taas lasketaan ennustetta käyttäjälle u toisesta ohjelmasta j ja senkin ohjelman on katsonut vain yksi käyttäjä u', tällöin

$$\begin{aligned} \boldsymbol{P}_{uj} &= \frac{\sum_{u'} \boldsymbol{W}_{uu'} \boldsymbol{Y}_{u'j}}{\sum_{u'} |\boldsymbol{W}_{uu'}|} \\ &= \frac{\boldsymbol{W}_{uu'} \boldsymbol{Y}_{u'j}}{|\boldsymbol{W}_{uu'}|} \end{aligned}$$

josta seuraa

$$\boldsymbol{P}_{uj} = \begin{cases} \boldsymbol{Y}_{u'j} & \text{jos } \boldsymbol{W}_{uu'} > 0 \\ -\boldsymbol{Y}_{u'j} & \text{jos } \boldsymbol{W}_{uu'} < 0 \\ \text{Ei määritelty} & \text{jos } \boldsymbol{W}_{uu'} = 0 \end{cases}$$

Tuloksista havaitaan, että käyttäjäpohjaisessa algoritmissa 2 ennusteet $\boldsymbol{P}_{ui}$ ja $\boldsymbol{P}_{uj}$ saavat toisistaan poikkeavia arvoja olettaen, että $\boldsymbol{W}_{uu'} \neq 0$.

Ohjelmasarjoja on yhteensä 1616, joista 219 eli 14 % on sellaisia, joita on katsonut vain yksi käyttäjä. 1616 ohjelmasarjasta 86 prosentilla on useampi kuin yksi katsoja, jolloin niiden osalta laskettavat ennusteet eivät pohjaudu vain yhden käyttäjän arvioon. Tällöin 86 % ohjelmasarjoista saadaan aina järjestettyä suosituimmuusjärjestykseen. Lopuissa 14 % ohjelmasarjoista, joilla on ollut vain yksi käyttäjä, ennusteeksi tulee edellä mainituista kahdesta vaihtoehdosta sama arvio vain niissä tilanteissa, joissa sama käyttäjä u' on ollut kahden arvioitavan ohjelmasarjan i ja i' ainoa arvioija. Käyttäjäpohjaisessa algoritmissa siis käyttäjälle laskettavista ohjelmasarjojen ennusteista vähintään 86 % pystytään aina järjestämään johonkin järjestykseen. Lisäksi järjestyksen oikeellisuus on sitä luotettavampi, mitä useampi katsoja on ohjelmasarjalla ollut.

Johtopäätökset

Kuten johdannossa mainittiin, Lasten Areenassa ei ole tällä hetkellä käytössä automaattista suosittelua. Tämän tutkielman puitteissa sellaista ei ole myöskään tuotettu Lasten Areenan käyttöliittymään saakka. Kun testijakson käyttäjä on päättänyt

katsoa jotain lastenohjelmaa, hän on tehnyt sen päätöksen näkemättä suositteluja. Siten mittaustulokset eivät heijastele suosittelun vaikutusta käyttäjien toimintaan. Toisin sanoen, mittaustuloksista ei voida päätellä, johtiko sisältösuositus sisällön kuluttamiseen vai ei. Tutkielman mittauksissa herkkyys- ja nDCG-mittarien tuloksia nostivat ne tapahtumat, joissa lastenohjelmien katsoja oli jo suosittelua näkemättä päättänyt katsoa jonkin ohjelman.

Tämä niin sanottu offline-testi mittaa kuitenkin sitä, että ennustaako suosittelualgoritmi oikein sen, mitä käyttäjä tulisi toistamaan Areenasta. Tällöin offline-testin tulosten avulla suosittelumallit ja -algoritmit voidaan järjestää keskenään paremmuusjärjestykseen. Tulosten perusteella tarjolla olevista suosittelumenetelmistä sekä osuvinta että monimuotoisinta suosittelua tarjoaisi Areenan suosittelujärjestelmä kaikilla toistotiedoilla opetettuna. Mikäli käytetään muistipohjaisia algoritmeja, lähdetietojen muoto vaikuttaa suuresti siihen, minkälaista suosittelua kukin algoritmi ja konfiguraatio tuottaa, jolloin oikean algoritmin valinta käytännön testein on tärkeää.

7 Yhteenveto

Suosittelujärjestelmien tarkoituksena on auttaa käyttäjää löytämään hänelle relevanttia sisältöä runsaasta sisältötarjonnasta [12]. Suosittelujärjestelmät tarjoavat käyttäjälle henkilökohtaisesti räätälöityjä sisältösuosituksia. Niiden käytön tavoitteena on säästää käyttäjän aikaa etsimiseltä. Sisältöjen tarjoajan näkökulmasta suosittelujärjestelmien käytöllä tavoitellaan myynnin edistämistä, monipuolisuuden korostamista sekä tyytyväisempiä ja uskollisempia käyttäjiä.

Lapset ovat sisältösuositusten käyttäjinä moninainen joukko. Lapsen kehitysvaiheet jakautuvat neljään vaiheeseen [14], jotka kuvaavat sitä, miten lapsen motoriset ja kognitiiviset kyvyt kehittyvät ikävuosien karttuessa. Lapset siis muodostavat etenkin tästä syystä varsin heterogeenisen ryhmän. Lasten suosittelussa on useampia sidosryhmiä, joiden tarpeita tulisi tyydyttää [15]. Lapsen vanhemmilla tai huoltajalla on näkemys lapselle sopivasta tai hyödyllisestä sisällöstä. Lapsen koululla tai lastentarhalla on myös omia näkemyksiään, minkälainen sisältö edistää lapsen kehittymistä ja oppimista. Yleisradiossa julkisen palvelun yhtiönä lastenohjelmistossa keskeisiä arvoja ovat lapsen kasvun tukeminen, suomalainen kulttuuri ja monipuolisuus [38]. Nämä arvot luonnollisesti vaikuttavat myös siihen, miten sisältöjen suosittelua lapsille toteutetaan.

Suosittelujärjestelmätyypeistä yleisimpiä on yhteistoiminnallinen suosittelu (engl. collaborative filtering) [12]. Muita tyyppejä ovat muun muassa sisältöön perustuva suosittelu. Yhteistoiminnallisessa suosittelussa lähdetietona on matriisi $\boldsymbol{Y}$, jossa on rivi käyttäjää kohden ja sarake sisältöobjektia kohden. Matriisin alkioiden arvoina on käyttäjä- ja sisältöobjektikohtaisia arvioita niistä sisältöobjekteista, joita kyseinen käyttäjä on arvioinut. Matriisi on siis sisällöltään harva, eli vain pieni osa matriisin käyttäjä- ja sisältöobjektikohtaisista alkioista on täytetty. Yhteistoiminnallinen suosittelu jakautuu kahteen ryhmään: muistipohjainen suosittelu ja mallipohjainen suosittelu [8]. Muistipohjaisessa suosittelussa sisältösuositukset käyttäjälle lasketaan käyttämällä kaikkia matriisin $\boldsymbol{Y}$ tietoja kerralla. Muistipohjaisessa suosittelussa suosituksia tuotetaan vertailemalla joko käyttäjien samankaltaisuuksia tai sisältöobjektien samankaltaisuuksia. Mallipohjaisessa suosittelussa luodaan malli, jonka avulla ennustetaan, miten käyttäjä arvioisi sisältöobjektin, jota hän ei ole vielä kuluttanut [23]. Toisin sanoen mallipohjaisessa suosittelussa ennustetaan matriisin $\boldsymbol{Y}$ tyhjien alkioiden arvoja. Malli koostuu käyttäjä- ja sisältöobjektikohtaisista vektoreista, joiden alkioita kutsutaan komponenteiksi. Vektoreista koostetaan käyttäjämatriisi $\boldsymbol{U}$ ja sisältöobjektimatriisi $\boldsymbol{I}$, jotka vuorostaan toimivat matriisihajotelmana täytetyn matriisin $\boldsymbol{Y}$ ennusteelle.

Tässä tutkielmassa tutkittiin kahden suosittelujärjestelmän suorituskykyä. Ensimmäinen tutkittava kohde oli Yle Areenan suosittelujärjestelmä konfiguroituna suosittelemaan pelkästään lastenohjelmia. Toisena tutkimuskohteena oli tätä tutkielmaa varten tehty muistipohjainen suosittelujärjestelmä.

Suorituskyvyn mittareina käytettiin herkkyys-mittaria (engl. Recall) [18], normalisoitua diskontattua kumulatiivista arvoa (engl. Normalized Discountive Cumulative Gain, nDCG) [21, 4] sekä käänteistä Simpson-indeksiä (engl. Inverse Simpson Index) [22, 34]. Herkkyyden ja nDCG:n avulla mitattiin suosittelun osuvuutta eli sitä, kuinka hyvin suosittelumallit ennustavat sitä, mitkä ohjelmasarjat olisivat käyttäjälle hyödyllisiä tai mieluisia. Työssä esiteltiin suosittelun monimuotoisuuden mittaamiseen ekologiasta tutumpi käänteinen Simpson-indeksi. Koska Yleisradiolla on julkisen palvelun yhtiönä velvollisuus tarjota monipuolista sisältöä, monimuotoisuuden mittaaminen esimerkiksi Ylen käyttämissä suosittelujärjestelmissä on perusteltua. Suosittelun suorituskykymittaukset tehtiin myös muun muassa satunnaisia suosituksia suosittelevasta suosittelijasta. Satunnaisen suosittelijan tulos oli sellainen, jonka suosittelualgoritmin tuli ylittää, jotta suosittelun voitiin katsoa toimivan.

Areenan suosittelujärjestelmästä tutkittiin kaksi pääasiallista konfiguraatiota. Ensimmäinen konfiguraatio oli kaikella toistotiedolla opetettu malli, jossa käyttäjälle näytettiin suosituksia esitettäessä vain lastenohjelmien suosituksia. Toinen konfiguraatio oli vain lastenohjelmien toistotiedoilla opetetut mallit. Vain lastenohjelmilla opetetuissa malleissa kokeiltiin erilaisten komponenttimäärien vaikutusta suorituskykyyn. Tuloksista ilmeni, että kaikella toistotiedolla opetettu malli oli sekä suosittelun osuvuuden että monipuolisuuden suhteen parempi kuin mikä tahansa lastenohjelmien toistotiedoilla opetetuista malleista. Lastenohjelmilla opetetuista malleista parhaat tulokset sai eniten komponentteja sisältävä malli. Tässä mallissa komponentteja oli käytössä 60. Vähemmän komponentteja käyttävät mallit saivat komponenttimäärää pienennettäessä vastaavasti pienempiä arvoja sekä herkkyydellä että normalisoidulla diskontatulla kumulatiivisella arvolla mitattuna.

Muistipohjaisesta suosittelijasta tutkittiin sekä käyttäjäpohjaista suosittelua (engl. user-based recommendation) että sisältöobjektipohjaista, tai tässä tutkielmassa ohjelmapohjaista suosittelua (engl. item-based recommendation). Käyttäjäpohjaisessa suosittelussa saatiin satunnaiseen suosittelijaan nähden selvästi parempia tuloksia. Tulokset eivät kuitenkaan yltäneet Areenan suosittelujärjestelmän tasolle. Ohjelmapohjainen suosittelu ei Areenan toistotietojen rakenteesta johtuen tuottanut satunnaista suosittelua parempaa suosittelua.

Tuloksista havaittiin, että Areenan suosittelujärjestelmä kaikkien ohjelmien toistotiedoilla opetettuna tuottaa näiden mittausten perusteella näistä suosittelutavoista

parhainta suosittelua. Tämän tutkimuksen perusteella erityisen mallin tai konfiguraation tuottaminen pelkästään lasten suositteluun ei ole tarpeen ainakaan suosittelun osuvuuden näkökulmasta. Mikäli haluttaisiin mitata suosittelun osuvuutta sidosryhmien näkökulmasta, tästä täytyisi järjestää oma erillinen tutkimus.

Jatkotutkimuksissa seuraava looginen askel olisi suosittelualgoritmien suorituskyvyn testaaminen online-testillä. Online-testillä tässä yhteydessä tarkoitetaan testaustapaa, jossa suosittelualgoritmi tuottaa suositukset aitojen käyttäjien nähtäville. Tällöin käyttäjän näkemien suositusten on mahdollista vaikuttaa hänen päätöksiinsä ja sitä kautta mittaustuloksiin. Online-testissä Lasten Areenan käyttäjät tulisi jakaa kahteen ryhmään: niihin, jotka näkevät suositukset ja niihin, jotka eivät näe suosituksia. Tällöin pystytään vertailemaan sitä, minkälainen vaikutus suosituksilla on käyttäjien toimintaan, eli missä määrin suositusten näkeminen lisää niiden käyttöä.

Toisena mahdollisena jatkotutkimuskohteena tulisi selvittää, mitä muita tietoja käyttäjistä voidaan kerätä ja voidaanko näitä tietoja käyttää tuottamaan entistä osuvampaa suosittelua Lasten Areenan käyttäjille. Eri ikäiset lapset ovat median kuluttajina varsin erilaisia [14]. Nykyisessä suosittelussa ollaan matriisihajotelman toimivuuden varassa sen suhteen, jotta esimerkiksi 4-vuotiaalle suositellaan hänen ikäiselleen sopivia ohjelmia. Tarkemman tiedon saaminen Lasten Areenan käyttäjän iästä saattaisi parantaa suosittelun suorituskykyä. Tarkemmat tiedot saattaisivat olla avuksi myös kylmäkäynnistysongelmassa, joka Areenan käyttötavoista johtuen on Areenassa relevantti ongelma.

Kuten luvussa 5.4 todettiin, Areenan suosittelujärjestelmässä tietojen käsittelyyn ja mallin laskentaan kuluu aikaa hieman yli tunti silloin, kun opetuksessa käytettävistä toistotiedoista on suodatettu pois muut kuin lastenohjelmat. Kaikkia toistotietoja käytettäessä aikaa kuluu hieman yli kolme tuntia. Joten, kun malli opetetaan vain lastenohjelmien toistotiedoilla, mallin laskenta sujuu noin kolmanneksessa siitä ajasta, minkä kaikilla toistotiedoilla oleva malli vaatisi. Mikäli laskentaan kuluvaa aikaa pidetään rajallisena resurssina, tällöin saman kolmen tunnin aikaikkunan puitteissa olisi mahdollista käyttää kuuttakymmentä suurempaa komponenttimäärää silloin, kun toistotiedoista on suodatettu pois muut kuin lastenohjelmat. Luvussa 6 tultiin siihen tulokseen, että komponenttimäärää kasvatettaessa suosittelun tarkkuus yleisesti ottaen paranee. Tämän vuoksi kolmantena jatkotutkimuskohteena tulisi vertailla lastenohjelmilla opetettujen mallien suorituskykyä 60 komponenttia suuremmalla komponenttimäärällä.

Lähteet

1 Laki Yleisradio Oy:stä 22.12.1993/1380. https://www.finlex.fi/fi/laki/ajantasa/1993/19931380, 1993. [haettu 1.12.2018].

2 Amatriain, Xavier and Basilico, Justin. Netflix recommendations: Beyond the 5 stars (part 1). https://medium.com/netflix-techblog/netflix-recommendations-beyond-the-5-stars-part-1-55838468f429, 2012. [haettu 28.1.2019].

3 C. Anderson. The long tail. *Wired magazine*, 12(10):170–177, 2004.

4 L. Baltrunas, T. Makcinskas, & F. Ricci. Group recommendations with rank aggregation and collaborative filtering. Teoksessa *Proceedings of the fourth ACM conference on Recommender systems*, sivut 119–126. ACM, 2010.

5 M. Banko & E. Brill. Scaling to very very large corpora for natural language disambiguation. Teoksessa *Proceedings of the 39th annual meeting on association for computational linguistics*, sivut 26–33. Association for Computational Linguistics, 2001.

6 M. W. Berry, M. Browne, A. N. Langville, V. P. Pauca, & R. J. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational statistics & data analysis*, 52(1):155–173, Elsevier, 2007.

7 L. Bottou. Online algorithms and stochastic approximations. Teoksessa *Online Learning and Neural Networks*, D. Saad, toimittaja. Cambridge University Press, Cambridge, UK, 1998. Tarkistettu, lokakuu 2012.

8 J. S. Breese, D. Heckerman, & C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. Teoksessa *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, sivut 43–52. Morgan Kaufmann Publishers Inc., 1998.

9 E. Brynjolfsson, Y. Hu, & D. Simester. Goodbye pareto principle, hello long tail: The effect of search costs on the concentration of product sales. *Management Science*, 57(8):1373–1386, INFORMS, 2011.

10 E. Brynjolfsson, Y. Hu, & M. D. Smith. Consumer surplus in the digital economy: Estimating the value of increased product variety at online booksellers. *Management Science*, 49(11):1580–1596, INFORMS, 2003.

11 E. Brynjolfsson, Y. J. Hu, & M. Smith. From niches to riches: Anatomy of the long tail. *Sloan Management Review*, 47(2):67–71, Heinäkuu 2006.

12 R. Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, Springer, 2002.

13 O. Chapelle & L. Li. An empirical evaluation of thompson sampling. Teoksessa *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, & K. Q. Weinberger, toimittajat, sivut 2249–2257. Curran Associates, Inc., 2011.

14 Y. Deldjoo, C. Frà, M. Valla, A. Paladini, D. Anghileri, M. A. Tuncil, F. Garzotta, & P. Cremonesi. Enhancing children's experience with recommendation systems. Teoksessa *Workshop on Children and Recommender Systems (KidRec'17)-11th ACM Conference of Recommender Systems*, elokuu 2017.

15 M. D. Ekstrand. Challenges in evaluating recommendations for children. Teoksessa *Workshop on Children and Recommender Systems (KidRec'17)-11th ACM Conference of Recommender Systems*, elokuu 2017.

16 D. Fleder & K. Hosanagar. Blockbuster culture's next rise or fall: The impact of recommender systems on sales diversity. *Management science*, 55(5):697–712, INFORMS, 2009.

17 P. Gopalan, J. M. Hofman, & D. M. Blei. Scalable recommendation with hierarchical poisson factorization. Teoksessa *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, UAI'15, sivut 326–335, AUAI Press, Arlington, Virginia, United States, 2015.

18 J. L. Herlocker, J. A. Konstan, L. G. Terveen, & J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, ACM, 2004.

19 Y. Hu, Y. Koren, & C. Volinsky. Collaborative filtering for implicit feedback datasets. Teoksessa *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, sivut 263–272. Ieee, 2008.

20 ISO 800000-2. Quantities and units – Part 2: Mathematical signs and symbols to be used in the natural sciences and technology. Standard, International Organization for Standardization, Geneva, CH, Dec. 2009.

21 K. Järvelin & J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, ACM, 2002.

22 L. Jost. Entropy and diversity. *Oikos*, 113(2):363–375, Wiley Online Library, 2006.

23 Y. Koren, R. Bell, & C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, IEEE, 2009.

24 X. N. Lam, T. Vu, T. D. Le, & A. D. Duong. Addressing cold-start problem in recommendation systems. Teoksessa *Proceedings of the 2nd international conference on Ubiquitous information management and communication*, sivut 208–211. ACM, 2008.

25 R. J. Larsen & M. L. Marx. *An introduction to mathematical statistics and its applications; 5th ed.* Prentice Hall, Boston, MA, 2012.

26 B. Lika, K. Kolomvatsos, & S. Hadjiefthymiades. Facing the cold start problem in recommender systems. *Expert Systems with Applications*, 41(4):2065–2073, Elsevier, 2014.

27 K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

28 D. W. Oard, J. Kim, et al. Implicit feedback for recommender systems. Teoksessa *Proceedings of the AAAI workshop on recommender systems*, vuosikerta 83. AAAI, 1998.

29 M. J. Pazzani & D. Billsus. Content-based recommendation systems. Teoksessa *The adaptive web*, sivut 325–341. Springer, 2007.

30 P. Resnick & H. R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, ACM, 1997.

31 F. Ricci, L. Rokach, & B. Shapira. *Recommender Systems Handbook*. Springer Publishing Company, Incorporated, 2nd edition, 2015.

32 B. Sarwar, G. Karypis, J. Konstan, & J. Riedl. Item-based collaborative filtering recommendation algorithms. Teoksessa *Proceedings of the 10th international conference on World Wide Web*, sivut 285–295. ACM, 2001.

33 M. Schedl & C. Bauer. Online music listening culture of kids and adolescents. Teoksessa *Workshop on Children and Recommender Systems (KidRec'17)-11th ACM Conference of Recommender Systems*, elokuu 2017.

34 E. H. Simpson. Measurement of diversity. *nature*, 163:688, Nature Publishing Group, 1949.

35 G. Takács, I. Pilászy, B. Németh, & D. Tikk. Major components of the gravity recommendation system. *Acm Sigkdd Explorations Newsletter*, 9(2):80–83, ACM, joulukuu 2007.

36 E. Tan, I. Seaman, H. Leung, & Y.-K. Ng. Making personalized movie recommendations for children. Teoksessa *Proceedings of the 18th International Conference on Information Integration and Web-based Applications and Services*, sivut 96–105. ACM, 2016.

37 Yleisradio Oy. Etusivu uusiksi! `https://yle.fi/aihe/artikkeli/2017/02/27/etusivu-uusiksi`, 2017. [haettu 21.8.2018].

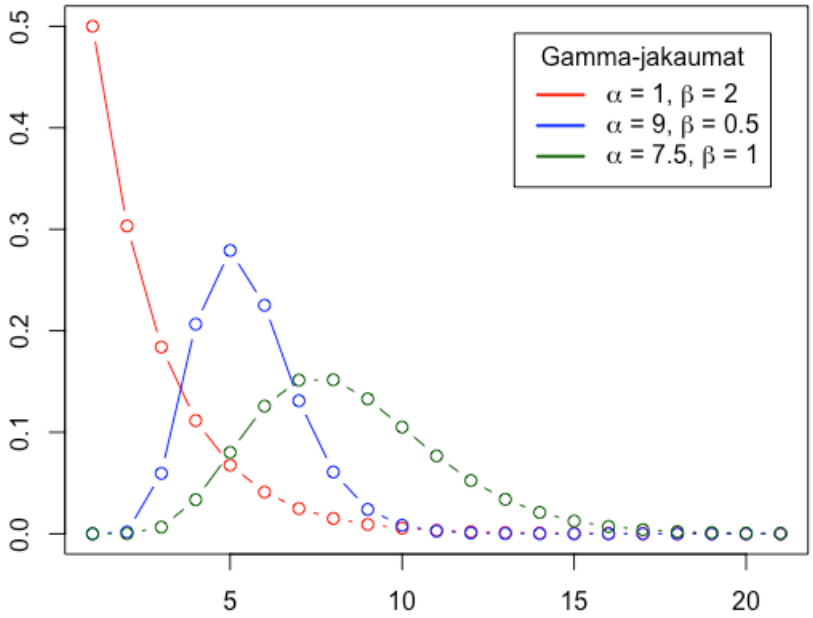
38 Yleisradio Oy. Lastenohjelmien periaatteet. `https://yle.fi/aihe/lapset/lastenohjelmien-periaatteet`, 2017. [haettu 27.8.2018].

39 Yleisradio Oy. Netti-tv: viikon 50 käynnistetyintä netti-tv-ohjelmaa yle areena, viikko 39/2018. `https://www.finnpanel.fi/tulokset/nettitvon/vko/topkaynn/viimeisin/topkaynn.html`, 2018. [haettu 6.10.2018].

40 Y. Zhou, D. Wilkinson, R. Schreiber, & R. Pan. Large-scale parallel collaborative filtering for the netflix prize. Teoksessa *International Conference on Algorithmic Applications in Management*, sivut 337–348. Springer, 2008.

Liite 1. Gamma-jakauma

Gamma-jakauma on kahden parametrin todennäköisyysjakauma [25]. Kuvassa 30 nähdään erilaisia gamma-jakaumien tiheysfunktioiden kuvaajia. Kuvaajassa vaaka-akseli kuvaa satunnaismuuttujan arvoa ja pystyakseli todennäköisyyttä.



Kuva 30: Gamma-jakaumien tiheysfunktioiden kuvaajia eri parametriparein.

Jos satunnaismuuttuja X on gamma-jakautunut, merkitään

$$X \sim Gamma(\alpha, \beta).$$

Gamma-jakauman tiheysfunktio on määritelty seuraavasti:

$$f_X(x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x},$$

missä $\Gamma(\alpha)$ on gamma-funktio, jonka määritelmä on seuraava:

$$\Gamma(\alpha) = \int_0^\infty x^{\alpha-1} e^{-x} dx.$$

Liite 2. Poisson-jakauma

Poisson-jakauma on yhden parametrin todennäköisyysjakauma [25].

Jos satunnaismuuttuja X on Poisson-jakautunut, merkitään

$$X \sim Poisson(\lambda).$$

Poisson-jakauman pistetodennäköisyysfunktio on määritelty seuraavasti:

$$P(X = k) = e^{-\lambda}\frac{\lambda^k}{k!},$$

jossa $k \in \mathbb{N} \cup \{0\}$.

Liite 3. Thompson-satunnaisotanta

Thompson-satunnaisotanta (engl. Thompson sampling) on eräs ratkaisu tutkimus-hyödyntäminen -ongelmaan (engl. exploration-exploitation) [13].

Oletetaan, että on olemassa jokin määrä peliautomaatteja, ”yksikätisiä rosvoja”, jotka antavat rahapalkkion jonkin tuntemattoman todennäköisyysjakauman mukaan. Tavoitteena on pelata näillä peliautomaateilla siten, että voitot saadaan ajan kuluessa maksimoitua. Tavoitteena on siis koettaa tasapainoilla sen suhteen, että valitaanko se peliautomaatti, joka tuntuu tarjoavan voittoja, vai valitaanko voittojakaumaltaan tuntemattomampia automaatteja suurempien voittojen toivossa.

Intuitiivisesti Thompson-satunnaisotanta tapahtuu siten, että oletetaan aluksi jokin todennäköisyysjakauma kullekin peliautomaatille. Yhtä peliautomaattia kokeillaan, jolloin saadaan sen automaatin antama palkkio. Tämän palkkion perusteella päivitetään käsitystä tuon peliautomaatin todennäköisyysjakaumasta. Seuraavaksi kokeillaan sitä peliautomaattia, jonka oletettu todennäköisyysjakauma on suotuisin. Prosessia jatketaan samaan tapaan, jolloin ajan myötä tarkentuu käsitys kunkin peliautomaatin todennäköisyysjakaumasta.

Formaalisti: oletetaan, että kontekstissa i valitaan toiminta $a \in A$, jonka jälkeen vastaanotetaan palkkio r. Menneistä toiminnoista muodostuu monikko $D = (x_i, a_i, r_i)$ Palkkion todennäköisyys on tällöin $P(r|a, x, \theta)$, riippuen tuntemattomasta parametrista θ. Parametrille θ on priorijakauma $P(\theta)$. Tällöin posteriorijakauma $P(\theta|D) \propto P(r_i|a_i, x_i, \theta)P(\theta)$.

Tällöin toiminta a valitaan todennäköisyydellä

$$\int \mathbb{I}\left[\mathbb{E}(r|a, x, \theta) = \max_{a'} \mathbb{E}(r|a', x, \theta)\right] P(\theta|\mathcal{D})d\theta.$$

Integraalia ei käytännössä tarvitse laskea erikseen. Riittää, että poimitaan satunnainen parametri θ kullakin Thompson-satunnaisotannan kierroksella.

Algoritmi 1 esittelee Thompson-satunnaisotannan pseudokoodina. T on suoritettavien kierrosten kokonaismäärä, jolloin t merkitsee sen hetkistä kierrosta. Alussa vastaanotetaan konteksti x_t, jonka jälkeen otetaan näyte parametrista θ posteriorijakauman $P(\theta|D)$ perusteella. Sen jälkeen valitaan toiminta a siten, että maksimoidaan palkkion odotusarvo. Lopuksi tehdään havainto palkkiosta r ja päivitetään menneiden tapahtumien joukkoa D.

Algoritmi 1 Thompson-satunnaisotanta [13]

1: **for all** $t = 1, ..., T$ **do**
2: Receive context x_t
3: Draw θ^t according to $P(\theta|D)$
4: Select $a_t = \arg\max_a \mathbb{E}_r(r|x_t, a, \theta^t)$
5: Observe reward r_t
6: $D = D \cup D(x_t, a_t, r_t)$
7: **end for**

Thompson-satunnaisotannasta ja sen suorituskyvystä on kerrottu tarkemmin artikkelissa [13].